Cognitive Models for Training Simulations

Annerieke Heuvelink

SIKS Dissertation Series No. 2009-24.

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Graduate School for Information and Knowledge Systems.
The research has been funded by - and conducted in cooperation with - TNO, the Netherlands Organization for Applied Scientific Research.

Thesis reading committee:
prof.dr. C. Castelfranchi (Institute of Cognitive Sciences and Technologies, Rome)
prof.dr. W.D. Gray (Rensselaer Polytechnic Institute, Troy)
prof.dr. F. van Harmelen (VU University Amsterdam)
dr.dr. J.F. Hoorn (VU University Amsterdam)
prof.dr. C.M. Jonker (Delft University of Technology)
prof.dr. J.-J. Ch. Meyer (Utrecht University)
prof.dr. M.A. Neerincx (Delft University of Technology / TNO Human Factors)

ISBN 9789086593484

VRIJE UNIVERSITEIT

Cognitive Models for Training Simulations

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad Doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. L.M. Bouter,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op vrijdag 11 september 2009 om 13.45 uur
in de aula van de universiteit,
De Boelelaan 1105

door

Annerieke Heuvelink

geboren te Rhenen

promotor:	prof.dr. J. Treur
copromotoren:	dr. K. van den Bosch
	dr. M.C.A. Klein

Contents

Chapter 1

Introduction

1.1 Motivation

Military organizations tend to operate in highly uncertain and dynamic environments, and therefore require competent staff that acts adequately in any emerging situation. Competent staff members are formed and maintained by training them in their task execution. However, the very nature of military missions makes it hard to set up real-world training. Practical issues are that the pace of military missions is often too low for training, while the level of danger is often too high. In addition, logistical issues play a role: mimicking a military mission in the real world requires many people and a large amount of money.

Fortunately, scenario-based simulator training is considered an appropriate alternative approach for training decision-making in complex environments (Oser, 1999). A main requirement for the success of simulator training is that it *correctly represents these aspects of the real world that are necessary to achieve the training objectives.* Which aspects these are varies from task to task. For example, when the simulated environment is used to train kite flying, it is important to represent wind and its influence on the kite, while these aspects are not required for training kite building.

1.1.1 Human Behavior Representation

The importance of validly representing the behavior of other humans in the simulated environment also varies between tasks. To train kite flying this is not important, nor is it for tasks in the process industry. But for a task like car driving, it is required that a person is trained in the interaction with road users in addition to the main driving task. Other tasks, like leading a team, solely consist of human interaction.

For military tactical decision-making the behavior of other humans, e.g., team members and opponents, is an important aspect. In order for simulator training to be an alternative for real-world training this behavior must be validly represented, i.e., have observational fidelity. Unfortunately, it can be hard to establish and hence to model which behavior is valid in any situation. To ensure that the simulated humans behave in a realistic way, Subject Matter Experts (SMEs) are often used to play these roles in tactical training scenarios. SMEs have the expertise to take the situational context into account and can use this understanding to representatively play a role in the training scenario. In addition, SMEs display varied behavior. This is required for training: when trainees are exposed to predictable behavior they might simply learn to 'play' with or against a specific entity, instead of learning their task. Moreover, SMEs are able to explain their behavior. This is a capability frequently used in the part of training called *after action review*, in which the trainee's behavior is critiqued.

Unfortunately, the logistical issues of real-world training, namely the large amount of money and organizational effort involved, also hold for simulator training when the attendance of SMEs is required. It would be highly beneficial when instead of SMEs computer programs could play roles in training simulations. This software should possess the capability of SMEs to respond in valid and varied ways to emerging situations.

1.1.2 Software Agents

Software that is capable of displaying autonomous behavior in interaction with other entities is generally referred to as an *agent*. This term is deduced from the Latin verb *agere*: to act. Agents are entities that exist in a world, and can observe, reason, and perform actions in that world (Russell and Norvig, 2003). What further constitutes agency has been the topic of much discussion (Franklin and Graesser, 1997).

Wooldridge and Jennings (1995) distinguish two general usages of the term agent and define two notions of agency. Because their first notion is relatively uncontentious, they call it the *weak* notion of agency. Their second notion is called *strong* as it is more contentious, and requires the software to satisfy additional constraints. The weak notion of agency defines an agent as a system that has the following properties: i) autonomy ii) social ability iii) reactivity and iv) pro-activeness. They state that for a system to fit into the stronger notion of agency it should further incorporate one or more concepts that are applicable to humans, for example: v) mentalist notions (beliefs, goals, plans, and intentions) vi) emotions vii) mobility viii) rationality or ix) adaptability.

Since no clear definition of agency exists that specifies its functionalities, many agent subclasses have emerged. Most distinguishable is the class of rational agents, which contains agents that only act in ways that help them achieve their goals and never in a

way that prevents this achievement. Although this property is very suited for developing software for a clear goal, it is not representative for human behavior.

1.1.3 Cognitive Biases

Task experts are expected to behave in a rational way. Unfortunately, there exist distinct and replicable ways in which human judgment and decision-making differs from decision-making based on rational choice (Tversky and Kahneman, 1974). It is generally acknowledged that these differences stem from the fact that human cognition has fundamental limitations (see, e.g., Miller, 1956; Kahnemann, 1973). These cognitive limitations force humans to apply simplified rules and heuristics while processing information for judging or decision-making. These simple rules often work well and are even regarded as adaptive given their ecological validity (Gigerenzer et al., 1999). However, when the outcome of such a simple rule deviates in a structural way from the rational outcome, it is called a cognitive bias.

Human decision-making is subject to a wide variety of cognitive biases (Wickens and Flach, 1988; Perrin et al., 1993). Such cognitive biases influence the quality of human decision-making and are found to arise especially under stress conditions (Baron, 2000). Military missions are generally stressful, and the decision-making processes of military experts are structurally affected by biases (Fewell and Hazen, 2005). It is therefore important to train military personnel in recognizing and dealing with their own biases, as well as with the biases displayed by their team mates.

1.1.4 Cognitive Models

For training military personnel in tactical decision-making in a simulated environment, it is important to validly represent the behavior of other humans present. Human behavior representation has a long history. Much work exists in Cognitive Science and Artificial Intelligence on the modeling of specific aspects of human behavior, such as vision, concept formation, rule learning, planning and motor control. Other work focuses on generic mechanisms and representation forms that may be useful for modeling multiple human behavioral aspects. Examples are logic engines, condition-action rules, neural nets and genetic algorithms (see, e.g., Russell and Norvig, 2003).

Several researchers have focused on determining the general characteristics of human behavior, with the goal to establish a so-called unified theory of cognition (UTC). A UTC is a single set of mechanisms that accounts for all aspects of cognition (Newell, 1990). These mechanisms are supposed to be constant over time, and across tasks and application domains. When these mechanisms are implemented in software they form a *cog-*

nitive architecture. Cognitive architectures constitute a fixed set of processes, memories and control structures that define their underlying theory about human cognition (Lewis, 2001).

Cognitive architectures can be used to build specific cognitive software agents. A cognitive software agent is formed by adding task-specific knowledge in the form of facts and rules to the cognitive architecture, which results in an executable cognitive agent model. Because theories of cognition differ, the behavior of a cognitive software agent is influenced by the architecture it is implemented in (Jones et al., 2007).

In this dissertation we define cognitive software agents as software agents with human-like cognitive capabilities. Note that this definition does not specify the required type of cognitive capability, or the way in which it should be implemented. Moreover, we treat executable agent models as synonymous to software agents: a cognitive (software) agent is equal to an *executable* cognitive agent model. A cognitive agent model always incorporates a cognitive model, but again, it is not specified how, or which type. A cognitive model by itself does not need to be executable, or form a complete agent model; many cognitive models only model a specific aspect of human behavior.

There is growing conviction and evidence that cognitive software agents can validly play roles within simulated environments instead of humans, and thus aid (military) training (Pew and Mavor, 1998; Ritter et al., 2003). However, much work remains to be done, e.g., on the incorporation of episodic memory in agents and on the reusability of the knowledge they embed (Langley et al., 2006).

1.1.5 Agent Requirements

In the previous sections we have implicitly discussed a number of requirements for cognitive software agents that play a role in a simulated training environment: they need to be able to show behavior that has observational fidelity, this behavior needs to vary, and they should be able to display biased behavior. In addition, it would be useful if they could explain their own behavior. These requirements directly correspond to the capabilities of Subject Matter Experts. However, the capability of SMEs to display varied behavior also contains a didactic disadvantage: the variability of the SMEs' behavior in combination with limited training time makes it hard to ensure that the trainee reaches all the training objectives, and therefore that he or she completes every stage of training.

It is desired that the behavior of simulated entities in training simulations is in service of the training goals. This is likely when instructors, who are SMEs that also possess didactic knowledge, play the roles required for a specific training scenario. Unfortunately, it is not realistic to require an instructor for each role. A second option would be that instructors could constrain the behavior of simulated entities to ensure that their behavior

is in service of the training objectives. Although it is hard to tune the behavior of humans, this is possible for cognitive software agents whose behavior is programmed.

Cognitive software agents that can validly represent human behavior offer a solution to the organizational effort involved in SMEs enabled simulator training. Cognitive agents whose varied behavior is tunable by an instructor offer a solution to the didactic issue of having SMEs play roles within a training scenario. To ensure cognitive agents also cut back expenses, the costs of developing them should not be too high. This entails that it should be avoided that for every new domain, task, or even simple scenario a new agent has to be built from scratch. Therefore, we take a component-based approach to designing cognitive agent models. This approach facilitates the reuse of the knowledge captured in the developed components.

1.1.6 Feedback Generation

The success of scenario-based simulator training depends not only on the valid representation of the relevant aspects of the task environment, but also on *the generation of feedback on the trainee's behavior* (Bosch and Riemersma, 2004). Usually human instructors monitor the trainee, evaluate the appropriateness of his or her behavior, and provide feedback. The use of instructors to generate feedback suffers from the same disadvantages as the use of SMEs or instructors to generate human behavior. Besides the logistic issues there is a didactic issue of variability in feedback between instructors. Although varied feedback is not necessarily harmful and possibly even useful, the military organization is keen on providing training in a structured way to ensure trainees get a comparable education.

It would be beneficial if software agents could not only replace human role players, but also human instructors. The latter is the focus of the research field on intelligent tutoring systems that combines knowledge and theories from Educational Science with methods from Artificial Intelligence. Intelligent tutoring systems comprise the knowledge of a domain expert, and use this knowledge to generate instructions and feedback to a trainee so he or she can learn about the domain (Polson and Richardson, 1988).

Most intelligent tutoring systems developed train procedural or simple declarative tasks. These tasks are typically well-defined, and feedback is based on expert knowledge in the form of rules or constraints that can unambiguously determine the correctness of certain actions in certain states. More challenging is the generation of feedback on trainee behavior in complex tasks like tactical decision-making. In such tasks the correctness of a certain action in a certain state can seldom be determined in a straightforward way. In general other aspects of the behavior should be taken into account for generating feedback, like the choices that were considered, or the sequential behavior.

Human instructors deal with the difficulty of diagnosing the task performance of a trainee by his or her visible actions by forming a mental model of the cognitive processes of the trainee. Subsequently, they base their feedback on this cognitive model. In order for a software agent to generate feedback on the (un)observable behavior of a trainee in a similar way, it should be able to reason about a trainee's cognitive processes. This can be done by forming a cognitive model of the trainee.

Furthermore, due to their years of training experience, instructors have a good sense of the kinds of errors trainees tend to make. Some of these errors are a direct result of cognitive biases. When the behavior of a trainee coheres with one of these wrong behaviors it is likely that the instructor classifies it as that specific error. Next, the instructor can provide feedback based on this match. In order for a software agent to recognize trainee behavior as a typical (cognitive) error, it should be able to reason about typical biased cognitive processes and their outcome. This can be supported by incorporating several (biased) cognitive models.

1.1.7 Synopsis

Although simulator training offers an alternative for real-world training of complex military tasks, currently such training depends on the availability of instructors and subject matter experts. The study presented in this dissertation aims at contributing to the development of (cognitive) software agents that can replace these humans. The agents that are to play a role in the simulated environment need to incorporate a cognitive model to do this in a valid, human-like way. The agents that are to give feedback on a trainee's task performance in a simulated environment need to be able to form and reason about cognitive models. Therefore this dissertation is titled *Cognitive Models for Training Simulations*. The ultimate goal is to replace all humans currently involved in simulator training, so that trainees can train by themselves at any time.

1.2 Research Objective

1.2.1 Research Focus

Many researchers share our interest in the modeling of human behavior in order to replace humans. However, this interest is not always focused on the modeling of the *cognitive* capabilities of humans. For example, most tasks at assembly lines are nowadays automated and performed by machines instead of humans. For such cases it is mainly important to model the human ability to perform specific manual operations. This dissertation focuses

on research that concerns the modeling of internal, cognitive aspects of humans and not on the modeling of external aspects, such as the visual system and motor behavior.

The modeling of cognition commonly services one of the following two goals. The first goal is to better understand human cognition. A model of a cognitive process is made with the aim to gain insight in the underlying mechanisms of that process. When the results and the behavior of the model strictly cohere with that of the cognitive process it aims to model, it can be deduced that the mechanisms underlying this process are accurately modeled. This method is also described as 'the logic of simulation' (Gilbert and Troitzsch, 1999), see Figure 1.1.

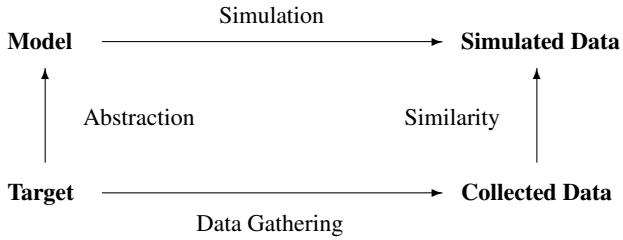


Figure 1.1: The logic of simulation as a method (Gilbert and Troitzsch, 1999)

The second goal of modeling cognition is to make artificial systems operate more intelligently, so that they can replace humans. For this it is not attempted to accurately model the mechanisms underlying cognition, but to capture specific characteristics. It is usually not the goal to develop a model that can replace humans in all aspects and in all circumstances: a model is developed to replace a human for a specific task. For that reason, the model does not need to represent all mechanisms underling human behavior, but only those mechanisms that are relevant for the particular task, and at a suitable level of abstraction.

These two separate goals closely follow the division made by March and Smith (1995) between *natural* and *design* science. Natural science is concerned with 'explaining how and why things are' and as such aims at understanding reality. Design science is concerned with 'devising artifacts to attain goals', and as such attempts to create things that serve human purposes.

This dissertation focuses on the development of software agents that can replace humans for a specific goal: for playing the role of a human in a training simulation, or for generating feedback on the task performance of a human trainee. This research can thus be viewed as a typical example of *design* science. Our aim for modeling cognition is not to understand it better, but to equip an artificial system, the software agent, with methods and techniques so that it can possibly replace a human.

1.2.2 Research Questions

The main research objective of this study is the modeling of cognitive agents that can generate human-like behavior for roles in training simulations of military tasks. We are interested in the modeling of a wide variety of human-like behavior, and do not only want to model behavior that can be considered expert behavior. In stressful situations, frequently faced by the military, behavior is often not rational due to the emergence of biases that influence decision-making. Biases do not always occur in the same form and amount: their occurrence is strongly affected by task circumstances. This makes it important to model biased behavior in training simulations, so a trainee can learn to recognize and deal with biases. Therefore, the main research question of this study is:

Q1: How can a cognitive agent display human-like behavior with a varying degree of biasedness?

This question embeds multiple aspects. First, it has to be established how an agent can display expert behavior. Next, it has to be determined which cognitive processes can become influenced by biases, how biases influence these processes, as well as when they do so. In the course of the research we narrowed this broad research question down to investigating whether it is possible to model human-like behavior with a varying degree of biasedness by extending qualitative models with quantitative elements.

In addition to this main research question, we have explored two supplementary research questions. It was mentioned that the modeling of the agents should happen efficiently to reduce their development costs, which spurs a component-based approach to the modeling of agents. In order to reuse components it is required that their properties are described in such a manner that at a later stage it can easily be determined whether they are suitable to be reused for a particular purpose. We hypothesize that it is a good idea to tag cognitive agent components with the cognitive capabilities they embed. However, there does not exist consensus on a taxonomy of cognitive capabilities, or on a way to describe them. Therefore, we start to investigate the following research question:

Q2: How can cognitive agent capabilities be described?

Besides the modeling of cognitive agents that display human-like behavior, we would also like to model agents that can generate feedback to a trainee in training simulations of military tasks. Previously, we explained that for an agent to generate feedback on trainee behavior in complex tasks, it needs to be able to reason about the cognitive processes of that trainee. This is required because the feedback for such tasks should be at the level of the cognitive processes of the trainee, and not solely on the factual outcome of his or her behavior. Therefore, we explore the research question:

Q3: How can an agent generate cognitive feedback on a trainee's task behavior?

In this dissertation we particularly investigate whether it is possible to generate cognitive feedback to a trainee on his or her task behavior by comparing it to the behaviors generated by expert and deficient task models.

1.3 Research Approach

Here we elaborate on the research approach followed in this study. In particular, we introduce the regulative research cycle, and apply it to our research objective.

1.3.1 Research Methodology

Previously, we introduced the work of March and Smith (1995) who divide research into *natural* and *design* science. March and Smith consider research to be *natural* science when it is concerned with explaining how and why things are, and as such aims at understanding reality. They view natural science as consisting of two activities: *discovery* (the process of generating or proposing scientific claims), and *justification* (includes activities by which such claims are tested for validity). These activities resemble phases within the methodological model known as the *empirical* cycle (Groot, 1969), which is commonly used to develop a theory within a dominant paradigm.

The empirical cycle includes the following phases: 1) *Observation*: empirical facts are collected; 2) *Induction*: hypotheses are formulated on the basis of the observed facts; 3) *Deduction*: on the basis of those hypotheses, some specific predictions are formed; 4) *Testing*: these predictions are empirically tested by collecting new data; 5) *Evaluation*: the results are evaluated on their theoretical validity. In this last phase new ideas are often generated that can be examined by a new empirical cycle.

March and Smith (1995) consider research to be *design* science when it is concerned with devising artifacts to attain goals, and as such attempts to create things that serve human purposes. They state that design science also consists of two basic activities, namely: *build* (the process of constructing an artifact for a specific purpose), and *evaluate* (the process of determining how well the artifact performs, which is complicated by the fact that performance is related to intended use). These two activities resemble phases within the methodological model known as the *regulative* cycle (Strien, 1997). This methodological model is more practice oriented, and focuses on solving an individual problem in particular circumstances.

The regulative cycle includes the following phases: 1) *Problem definition*: identification of a discrepancy between an actual and a normative situation; 2) *Diagnosis*: clear

formulation of the problem; 3) *Plan*: development of a solution for the identified problem; 4) *Intervention*: implementation of that solution; 5) *Evaluation*: testing whether the proposed solution has narrowed the gap between the actual and normative situation. The last phase may identify new or remaining problems that can be examined by a new regulative cycle.

The regulative cycle is normative in the sense that the development of a plan is guided by an objective derived from the problem under consideration. This makes it applicable to design-oriented research, and stresses what Simon (1967) already expressed:

> "The engineer and more generally the designer, is concerned with how things ought to be - how they ought to be in order to attain goals, and to function (...) With goals and 'oughts' we also introduce into the picture the dichotomy between normative and descriptive. Natural science has found a way to exclude the normative and to concern itself solely with how things are (...) Artificial things can be characterized in terms of functions, goals and adaptation."

The research described in this dissertation is a clear example of *design* science and therefore, the research method embodied by the regulative cycle is applicable.

Although the empirical and the regulative research cycle are presented as separate processes, they are inevitably connected. Theories that are the result of the empiric cycle are often used within the first phases of the regulative cycle. Because of this application of theory into practice, feedback from the intervention and evaluation phases of the regulative cycle can in return be used to further develop theories by the empirical cycle.

1.3.2 Regulative Research Cycle

The aim of our study is to contribute to the development of software agents that can fulfill tasks within simulator training of military tasks that are currently fulfilled by humans. This general research objective emerged from the first phase of the regulative cycle, namely the *problem definition* (1). In the first section of this chapter we identified a clear discrepancy between the actual situation in which trainees are trained for open and complex tasks, and the desired situation. The actual situation is that other humans are required for such training which induces high costs, great organizational effort, and undesired training variability. The desired situation is that trainees can train by themselves because the humans are replaced by software agents. For this, agents 1a) should be able to show valid behavior in a simulated task environment which can be tuned to be more or less biased, and 1b) need to be affordable and as such be based on components that can be found for reuse. In addition, agents 1c) should be able to give feedback on (biased) trainee behavior in simulated task environments.

The *diagnosis* (2) of the problem that becomes explicit throughout this dissertation splits the problem up in three main issues: no suitable methods exist for 2a) modeling various required behaviors of cognitive agents in a way that that behavior is valid, varied and has a varying degree of biasedness, nor for 2b) describing the developed agent components so that they can be found for reuse. Furthermore, 2c) no method exists that enables an agent to generate feedback on (biased) trainee behavior in open, dynamic, complex tasks.

Therefore our *plan* (3), reflected in the research questions listed in Section 1.2.2, is to 3a) develop methods with which various required behaviors of cognitive agents participating in a military simulation can be modeled, and to 3b) investigate how the capabilities of developed agent components can be described. Moreover, we plan to 3c) develop a method which enables an agent to generate feedback on trainee behavior for open, dynamic, complex tasks.

In phase (4), *intervention*, the developed methods are used to 4a) implement cognitive agents that show (biased) behavior within a simulated environment, to 4b) describe various capabilities of cognitive agents, and to 4c) implement an agent that generates feedback on trainee behavior in a simulated environment.

In the final *evaluation* (5) phase it is tested whether our study aids in narrowing the gap between the actual and normative situation. In other words: whether it contributes to the future modeling of agents that 5a) can show valid, human-like behavior with a tunable degree of biasedness in complex tasks in simulated environments, 5b) are affordable because they are based on components that can be found for reuse, and 5c) can give feedback on (biased) trainee behavior in complex tasks in simulated environments.

This regulative cycle is the major research cycle of the study described in this dissertation. In it multiple sub-cycles are embedded, each investigating a sub-problem of a non-existing method to model a required agent behavior (2a).

These regulative sub-cycles start in general with a *problem definition* in the form of: there exists a discrepancy between the aspects of human behavior that existing methods can model and the aspects that are required for this specific (military) task. To establish such a discrepancy, first the cognitive properties of human behavior that are required to validly fulfill that specific task are determined. At the same time it is determined which cognitive properties current approaches can model. When a discrepancy is found, the *diagnosis* states clearly which cognitive properties the to-be-developed method should be able to model. Next, in the *plan* phase, inspiration for a solution to the problem is drawn from theories about the underlying mechanisms of the required human behavior developed by natural science. The solution is first logically formalized and then, in the *intervention* phase, implemented for a specific case. These implementations vary from

executable models for simple abstract tasks, to models implemented in existing cognitive architectures for realistic tasks. Next, these agent implementations are *evaluated*. These evaluations vary due to the variety in implementations. They range from checking whether the agent's behavior displays the cognitive properties established in the diagnosis phase, to checking its face validity by consulting subject matter experts.

1.3.3 Related Research Disciplines

From this regulative research cycle it follows that many research disciplines are relevant to this study's research objectives. Below the major ones are listed, together with a short description of how they contribute.

- **Cognitive Science** - provides unified theories of cognition as well as many specific theories on cognitive processes like planning, belief revision, attention and stress.
- **Computer Science** - provides a computational means to formalize and test the theoretical models of cognitive science. Various unified as well as specific computational cognitive models have been developed. In addition, computer science provides methods to describe software and its working.
- **Artificial Intelligence** - provides various techniques for modeling intelligent behavior. AI draws its inspiration from human intelligence, however, its techniques are not necessarily cognitively valid or plausible.
- **Educational Science** - provides insight in how humans learn and offers guidelines concerning the kinds of training environments and types of feedback that are suited for training specific tasks.

1.4 Research Scope

In this section we discuss the scope of the study described in this dissertation. We start with the context in which the study was conducted, and then elaborate on the domain it focuses on. Next, we introduce the military task that is used throughout this study as example of which the training can be supported by software agents. As an outlook, this section lists the requirements for agents capable of executing or providing feedback to the example task.

1.4.1 Research Context

The study described in this dissertation was conducted in a cooperation between the Agent Systems Research group of the Vrije Universiteit Amsterdam and the Training

and Instruction department of TNO Human Factors. TNO Human Factors is a business unit of TNO Defense, Security and Safety, which is one of the five core areas of TNO, the Netherlands Organization for Applied Scientific Research. The study took place in parallel to the TNO research program **Cognitive Modeling** (V524), funded by the Netherlands Defense Organization. This program focuses on 'Cognitive Models of Tactical Decision-Making' and incorporates three projects: *Training*, *Decision Support* and *Agent Architectures*.

Within the *Training* project cognitive models are developed for training purposes; it is investigated whether such models can make training more realistic, more traceable, and more cost-efficient. The *Decision Support* project develops cognitive models for decision support; it investigates how such models can be used to deliver adaptive support, tailored to the operator's decision-making process. *Agent Architectures* is a coordinating project; it investigates and develops architectures required for implementing the cognitive models into intelligent agents, and for linking these agents to simulation systems.

Predating the Cognitive Modeling research program, the researchers within the Training and Instruction (T&I) department did not have much knowledge about the development of cognitive models and their implementation in software to form intelligent agents. Their main expertise was the training of people, especially in complex decision-making tasks. Researchers within the Agent Systems Research (ASR) group did not have experience with this type of training. However, they did have many years of experience with designing intelligent agents, and in more recent years also in cognitive modeling (Bosse, 2005). One of the accomplishments of the ASR group has been the development of component-based system design method DESIRE that explicitly models agents, their environment, and their interaction, during different phases of design (Brazier et al., 2002).

By conducting this study in a cooperation between a university and a research institute, the feedback cycle embedded in the regulative research cycle was fostered. In the previous section we introduced the regulative research cycle. We elaborated on how the application of theory into practice by the intervention phase can, through the final evaluation phase, lead to feedback on new or remaining problems that can be examined by a new regulative cycle. In this study, we mainly developed mechanisms and techniques in cooperation with the university, and subsequently implemented and evaluated them in cooperation with the research institute. The experiences gained were used to formulate new requirements for the cognitive agent models, which were succeedingly investigated by a new research cycle. As a result of the cooperation, the T&I department of TNO gathered knowledge concerning modeling (cognitive) agents, while the ASR group got inspired by the possibilities to use cognitive models to support humans, not only for training but also for decision support.

1.4.2 Research Domain

The research domain of the Cognitive Modeling program is *Tactical Command*, which is defined by the Oxford Essential Dictionary of the U.S. Military as:

> "The authority delegated to a commander to assign tasks to forces under his or her command for the accomplishment of the mission assigned by higher authority."

A commander 'assigning tasks to forces under his command to accomplish mission success' is executing a tactical decision-making process. A concept often applied to describe the decision-making process in military operations is the OODA loop, which stands for Observe, Orient, Decide and Act. The OODA loop, also called Boyd cycle, was developed by USAF Colonel John Boyd, and describes human decision-making as a recurring cycle of observe-orient-decide-act, see Figure 1.2.

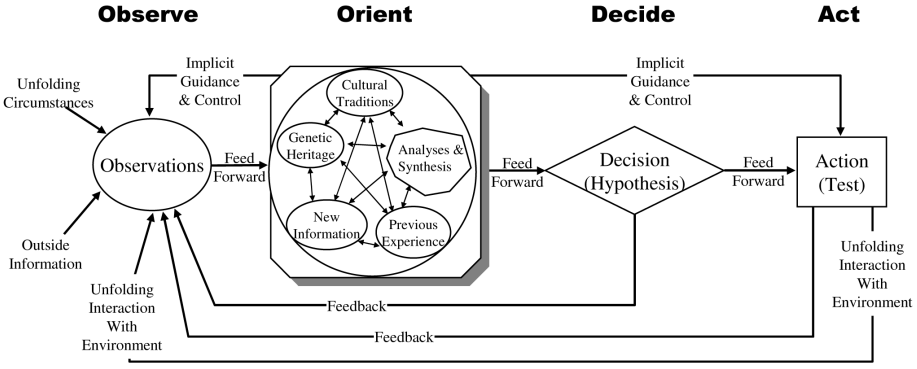


Figure 1.2: John Boyd's OODA loop (adapted from Boyd (1996))

The main focus of the research program and this study lies on the **Orient** part of the OODA-loop, which denotes the commander's assessment of the current situation. However, the modeling of the entire loop is investigated since orientation is intertwined with observing the environment, with making decisions, and with acting based on the assessed situation. In addition, all these processes need to be modeled to form an executable agent.

1.4.3 Research Task

The Orient part of the OODA-loop is aimed at achieving situational awareness (SA), which is a state of knowledge. Endsley (1995) defines SA as:

> "the perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future".

Another definition is given by the US-Army (2002) who defines SA as:

> "the ability to maintain a constant, clear mental picture of relevant information and the tactical situation. This picture includes knowledge of both the friendly and threat situations and of relevant terrain".

The process of achieving, acquiring, or maintaining SA is referred to as *situational assessment* (Endsley, 1995). It is generally recognized that lacking SA, or having inadequate SA, is one of the primary reasons for decision-making errors. Therefore, training students in achieving good SA, i.e., in the situational assessment task, is a spearhead of the Netherlands Defense Organization.

Military tactical experts achieve Situational Awareness in a quick and accurately way by using their large knowledge base of tactical patterns gained over time. Zsambok and Klein (1997) show that experts use this accumulated experience when making a decision: they base their decision on recognized pattern similarities between the actual decision-making situation and stored situations.

In order to train students to become tactical experts, they should be engaged in intensive practice facing a wide variety of situations. This enables them to build up experience in achieving SA, and to expand and sophisticate their knowledge base of tactical patterns. This is an important motivation for our requirement that the cognitive agents that are to display human-like behavior in training simulations, should be capable of displaying *varied* behavior.

TNO's Cognitive Modeling program is of interest to all three major defense areas: the Royal NetherLands Navy (RNLN), Army, and Air Force. However, for this study the naval domain was selected as main testbed. The Operational School of the RNLN (Opschool) is concerned with the training of tactical decision-making. Students master this task by learning tactical theory and practicing tactical decision-making. The training consists of repeated practice of tactical decisions in order to improve these decisions. For this training the Opschool uses a semi-automated system called the Action Speed Tactical Trainer (ASTT), which can simulate the command central of a military ship in a naval battle, see Figure 1.3.

For training tactical decision-making using the ASTT humans are required, because the reactions of opponents and other parties to the actions of the students have to be programmed during the exercise. The aim of our study it to develop methods and techniques for modeling intelligent, cognitive agents whose behavior does not need to be programmed, but that can *autonomously* act in response to the student. The Opschool was supportive of this goal: they delivered the domain experts required for the elicitation of task knowledge for some of the cognitive models of the agents developed, as well as for the validation of the behavior of these agents.

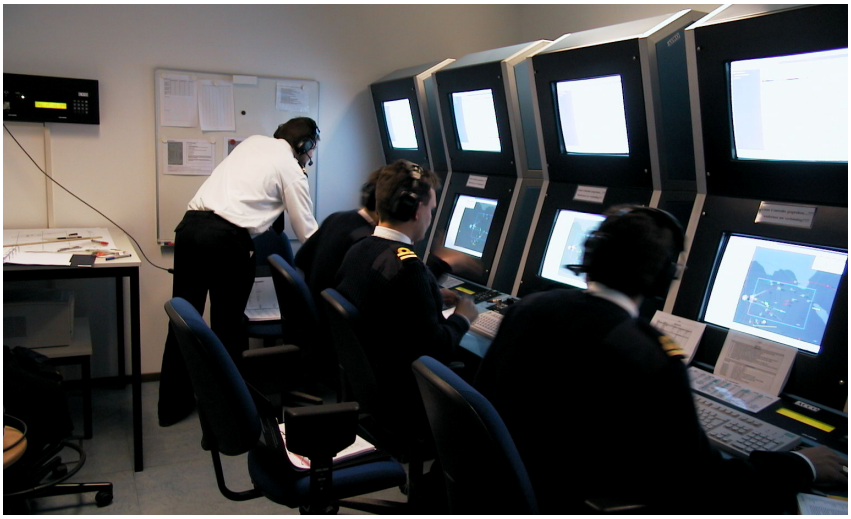


Figure 1.3: Students are trained in naval warfare using the Action Speed Tactical Trainer

1.4.4 Agent Requirements for the Research Task

In this study, we explicitly focus on the modeling of the cognitive processes involved in situational assessment.

Displaying Human-Like Behavior

For agents that participate in the simulation and should be able to perform the situational assessment task as, e.g., a team member or opponent, many aspects need to be modeled. The agent needs to have facilities to:

- observe relevant information in the simulated world;
- interpret the observed, possibly uncertain, information;
- represent that interpretation, e.g., in the form of a belief;
- store interpretated information, i.e., some kind of memory;
- retrieve information from this memory;
- integrate information from different sources or over time;
- infer new information (beliefs) from current information (beliefs);
- be pro-active, e.g., by incorporating goals;
- form and adapt goals based on new information;
- decide which of its goals to pursue;
- decide on, i.e., plan, actions to reach a goal;
- perform actions in the simulated world.

These listed processes are minimally required for situational assessment: they all need to be modeled to enable a software agent to execute that task. In addition, for modeling human-like behavior that is not necessarily rational, some of the following aspects should be modeled:

- how expectations influence the sensing of information or the formation of beliefs;
- how emotions influence the formation of beliefs or the decision on a course of action;
- how cognitive limitations influence the retrieval of beliefs or the execution of (heuristic) reasoning rules;
- how stress / exhaustion / workload has effect on these cognitive limitations and thereby influence the sensing of information, the formation of beliefs, or the decision of a course of action.

This list is not complete: more cognitive aspects, and a multitude of cognitive processes influenced by them can be listed. Fortunately, in order for a software agent to display human-like behavior, in the sense that that is not always rational, it is not required to model all these aspects and how they influence cognitive processes. The modeling of a specific aspect, e.g., expectations, can suffice to create biased behavior.

In the current study we focus on a subset of these aspects and leave others, like emotions, out. In particular, we examine the modeling of biased behavior by focusing on cognitive limitations, and on the circumstances under which these limitations especially bias behavior, e.g., when people are stressed or cognitively exhausted. Moreover, we investigate the influence of these aspects on a number, and not all, of the cognitive processes listed. In this dissertation, we start with the modeling of possibly biased behavior for the interpretation of observed information, for the integration of the resulting beliefs, and for the deduction of new beliefs from others. These processes are amongst the ones most relevant for executing the situational assessment task. Next, we focus on the modeling of possibly biased retrieval of information from memory, a process initially left out. Last, we model a control mechanism for possibly biased belief deduction, as well as control mechanisms for biased decision-making on actions in order to reach a goal.

Providing Feedback to Human Behavior

For providing feedback to a student performing the situational assessment task, an agent needs to be able to reason about the cognitive processes required to perform the task. In particular, an agent has to have access to expert task knowledge, and to knowledge about the type of errors that students typically make (instructor knowledge). In addition, it must be able to observe the behavior of the trainee, and to compare these observations with the behavior of the expert and with typical errors. After diagnosing the trainee's task

performance the agent should be able to provide feedback to the student. Last, it would be beneficial if it can store its diagnosis, so that over time it can also generate feedback on the overall task performance.

1.5 Dissertation Outline

This dissertation consists of eight chapters. Chapters 1, 2, and 8 are umbrella chapters in the sense that they respectively introduce, discuss related work to, and discuss and conclude the research that is described in Chapters 3 to 7. Each of these five chapters includes an introductory part and one to four papers. In this section we elaborate on the content of the chapters, and list the papers embedded in them.

1.5.1 Chapter Overview

In the current chapter, **Introduction**, we have discussed the general motivation for this study and listed the three research questions we aim to answer. In addition, we elaborated on the context of this study and on the military research domain, and introduced a specific military task as research example. Moreover, we introduced the regulative research cycle as the research methodology that this study follows. The chapter's first sections denote the *problem definition* of our main regulative cycle, and started with a formulation of the problem and the development of a solution for it.

Chapter 2, **Related Research**, discusses related research which helps to further *diagnose* the research problem. We elaborate on Cognitive Science as well as Artificial Intelligence research concerning the modeling of (biased) cognitive behavior. In addition, we discuss research concerning the generation of feedback.

Chapters 3 to 7 all embed a regulative sub-cycle. The chapters 3, 4 and 5 investigate the phases *a* of the main regulative cycle. Chapter 6 brings about phases *b*, while chapter 7 focuses on phases *c* of the main regulative cycle.

Chapter 3, **Belief Component**, starts with an introductory section in which we elaborate on the research problem, diagnosis, and plan concerning a cognitive agent's belief maintenance capability for situational assessment tasks. In the following section this plan is worked out which leads to a formal belief framework for cognitive agent models, which use is demonstrated in a simple case study (Heuvelink, 2007). Next, this framework is used to develop a formal task model of a realistic military task. This task model is implemented in the cognitive architecture ACT-R which results in the cognitive agent BOA (Heuvelink and Both, 2007) that is subsequently validated (Both and Heuvelink, 2007). The generality of the belief framework and task model are tested by a reimple-

mentation of the model in the Soar cognitive architecture, which results in the cognitive agent Boar (Muller et al., 2008).

Chapter 4, **Memory Component**, introduces a memory capability for an agent incorporating the developed belief framework of Chapter 3 enabling it to store, retrieve, and make inferences on its beliefs. For this, we first develop a method to perform arbitrary aggregations on these beliefs (Heuvelink et al., 2008b) which is later incorporated in the memory model. The memory model supports an agent in performing human-like (biased) reasoning as well as rational reasoning (Heuvelink et al., 2008a), among others by introducing an availability value for beliefs.

Chapter 5, **Control Component**, starts with an introductory section in which we discuss several control aspects for cognitive agents. In the following section we introduce a formal control method that determines the kind of reasoning behavior (biased versus rational) an agent shows. This is not necessarily fixed, but can change dynamically over time due to internal and external aspects (Heuvelink and Treur, 2008). In the next section, we elaborate on an information acquisition component that determines whether an agent senses or tries to remember required information (Heuvelink et al., 2009a). For this research we performed an experiment to deduce human information acquisition behavior in a simple task. The experimental data served as inspiration for the modeling of various task strategies, and was used to evaluate the developed information acquisition model.

Chapter 6, **Cognitive Agents Capabilities**, introduces our idea to facilitate the reuse of components of cognitive agents by tagging them with descriptions of the cognitive capabilities they embed (Heuvelink et al., 2009b). The preliminary Capability Description Framework (CaDeF) proposes a method to describe cognitive capabilities which is demonstrated by describing two generic ones: reasoning and decision-making. In addition, CaDeF is used to describe specific instantiations of these two generic capabilities in an implemented agent, namely BOA (Both and Heuvelink, 2007).

Chapter 7, **Feedback System**, introduces a multi-agent-based feedback generating system developed to generate cognitive feedback to the behavior of trainees in open, complex tasks. The diagnosis capacity of the feedback generating agent is evaluated by letting it diagnose student agents (Heuvelink and Mioch, 2008).

Chapter 8, **Conclusion**, sums up the research and discusses its relevance and significance in relation to the motivation and research questions established in the current chapter. In specific, we address the points of the *evaluation* phase, namely whether the research aids in the future modeling of agents that 5a) can show valid, human-like behavior with a tunable degree of biasedness in complex military tasks, 5b) are affordable because they are based on components that can be found for reuse, and 5c) can give feedback on (biased) trainee behavior in complex military tasks.

1.5.2 Embedded Papers

- Heuvelink, A. (2007). A belief framework for modeling cognitive agents. In *Proceedings of the 8th International Conference on Cognitive Modeling (ICCM 2007)*, pages 235–240. Psychology Press.

- Heuvelink, A. and Both, F. (2007). BOA: A cognitive tactical picture compilation agent. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, pages 175–181. IEEE-CS Press.

- Both, F. and Heuvelink, A. (2007). From a formal cognitive task model to an implemented ACT-R model. In *Proceedings of the 8th International Conference on Cognitive Modeling (ICCM 2007)*, pages 199–204. Psychology Press.

- Muller, T. J., Heuvelink, A., and Both, F. (2008). Implementing a cognitive model in ACT-R and Soar: A comparison. In *Proceedings of the 6th International Workshop on From Agent Theory to Agent Implementation (AT2AI-6 2008) in conjunction with AAMAS 2008*.

- Heuvelink, A., Klein, M. C. A., and Treur, J. (2008b). A formal approach to belief aggregation. In *Proceedings of the 12th International Workshop on Cooperative Information Agents (CIA 2008)*, volume 5180 of *Lecture Notes of Artificial Intelligence*, pages 71–85. Springer-Verlag.

- Heuvelink, A., Klein, M. C. A., and Treur, J. (2008a). An agent memory model enabling rational and biased reasoning. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2008)*, pages 193–199.

- Heuvelink, A. and Treur, J. (2008). Controlling biases in demanding tasks. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society (CogSci 2008)*, pages 1392–1397. Cognitive Science Society.

- *An extended version of the paper:* Heuvelink, A., Klein, M. C. A., and Lambalgen, R. L. C. v. (2009a). Modeling human information acquisition strategies. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society (CogSci 2009)*. Cognitive Science Society. *In print*.

- Heuvelink, A., Mioch, T., and Doesburg, W. A. v. (2009b). CaDeF: Towards a method for describing cognitive agent capabilities. *Unpublished*.

- Heuvelink, A. and Mioch, T. (2008). FeGA: a feedback generation agent. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2008)*, pages 567–572.

Chapter 2

Related Research

2.1 Introduction

The goal of this study is to contribute to the automation of military tactical simulator training, so that in the future trainees can train at any time and by themselves. In Chapter 1 we stated that for this type of training it is minimally required to validly represent human behavior in the simulated environment, and to provide feedback to the behavior of the trainee. As a possible solution to automate such training we introduced the notion of software agents: we proposed that instead of humans, agents can deliver these required behaviors. In addition, we put forward that software agents that are to display human-like behavior should incorporate a cognitive model. The aspects of cognition that this model needs to represent will depend on the task of the agent in the simulated environment and the goal of the training. Moreover, the software agents that are to generate feedback on the behavior of the trainee should be able to reason about the trainee's cognitive process, and can do this by means of cognitive models. Once more, the goal and task of the training will influence the processes the agent has to reason about, and thus the cognitive models required.

The study of cognition, the determination of its distinct capabilities, the investigation of these separate capabilities as well as the entire system by performing experiments and building models, is labeled *Cognitive Science*. The research conducted and described in this dissertation is closely related to this field of research. When developing cognitive models it has to be established which capabilities are required, how these should operate together, and how all this can be modeled. The research field of *Artificial Intelligence* also studies (aspects of) cognition, and determines and models its distinct capabilities as well as the entire system. However, its experiments do not investigate whether the

models validly represent human cognition, but if they display behavior at the level of intelligence required for the task they are developed for.

Taking into account the two goals of modeling cognition described in Section 1.2.1, Cognitive Science can be considered a natural science, while Artificial Intelligence is its design science counterpart. Our research draws inspiration from Cognitive Science, but ultimately belongs to Artificial Intelligence. We are concerned with the modeling of cognition, because we want to generate human-like behavior and feedback on human behavior, and not because we want to penetrate the processes underlying cognition.

Chapter Overview

In this chapter, we discuss research that is related to our research questions. The function of this chapter is to provide background knowledge on a wide variety of topics, to increase the readability of the following chapters for those that are not familiar with the topics addressed. Not only this chapter, but also the introduction sections of the coming chapters and of the papers they embed discuss related research. However, especially in the papers, these discussions already focus on addressing specific design issues, while here we aim to provide a generic overview of several research topics.

We start this chapter with discussing global aspects of cognition: in Section 2.2.1 we introduce various categorizations of capabilities that presumably constitute cognition. Next, in Section 2.2.2, we elaborate on the origin of suboptimal task performance by humans. This is an aspect of human behavior that our software agents should be able to display, and provide feedback about.

Then, we discuss how cognition could be modeled. In Section 2.3.1 we shortly discuss various approaches toward the modeling of cognition, i.e., using qualitative and/or quantitative terms, and introduce our approach to the modeling of cognition. In addition, in Section 2.3.2, we discuss the notion of integrated architectures. We address two cognitive architectures in detail: Soar (Laird et al., 1987) and ACT-R (Anderson and Lebiere, 1998; Anderson et al., 2004), and elaborate on their relevancy for our research objectives.

Next, we discuss possible application areas for cognitive models. In Section 2.4.1 we elaborate on a variety of application domains for which software agents have been developed with the purpose to generate human-like behavior. For each domain we discuss how the requirements of the behavior of those agents match the requirements of the behavior of an agent for our goal: a tactical training simulation. In Section 2.4.2, we elaborate on various methods to generate feedback, and discuss software models capable of generating feedback.

Finally, in Section 2.5, we conclude this chapter and look forward to the next.

2.2 Aspects of Cognition

What does it actually entail 'to be cognitive', i.e., what kind of processes does cognition embed? This is a hard question, since *cognition* is an umbrella notion. The Oxford English Dictionary (1994) defines cognition as:

> "The action or faculty of knowing taken in its widest sense, including sensation, perception, conception, etc., as distinguished from feeling and volition."

For the studying and modeling of cognition it is useful to divide this umbrella term in clear conceptual parts. In this section we start with the introduction of several proposals for the division of cognition. Next, we elaborate on the generic aspect of cognition that it is not always rational. We introduce several examples of cognitive biases, and discuss their (presumable) origin.

2.2.1 Cognitive Capabilities

When thinking about cognition it is easy to come up with specific cognitive processes that humans posses, ranging from selecting a filling for their sandwich, to determining the best time to call their grand-mother. Such processes can be categorized as being instances of specific cognitive capabilities: decision-making and scheduling, respectively. Many of these distinct capabilities have been identified and extensively researched within Cognitive Science and Artificial Intelligence, with as respective goals to understand and to model them.

In this section we will not elaborate on specific research projects that focused on the studying and modeling of a specific capability. The introduction sections of Chapters 3, 4, and 5 will, since these chapters focus on specific capabilities. Instead, we present three research projects that have attempted to provide an overview of cognitive capabilities. These projects serve as examples; it is not our goal to subscribe to one of these views. They are merely provided as illustrative for the number and level of capabilities that can be distinguished within human cognition.

Gordon (2005) states that there are sixteen functional requirements (capabilities) that a model of human cognition should possess. In Table 2.1 we present the taxonomy of the sixteen functional classes of cognitive models that Gordon proposes. Gordon bases this taxonomy on formal theories of 'commonsense psychology' and states that each of these functions must be encoded within an agent for it to be able to perform commonsense reasoning. Although not all these capabilities are required for the current task, it provides a good overview of the processes an agent should eventually be capable of in order to mimic human behavior in all its facets.

Table 2.1: The 16 Functional Classes of Cognitive Models according to Gordon (2005)

Functional Class	**Summary**
1. Knowledge and inference	Models of how people maintain and update their beliefs in the face of new information
2. Similarity judgment	Models of how people judge things to be similar, different, or analogous
3. Memory	Models of memory storage and retrieval
4. Emotion	Models of emotional appraisal and coping strategies
5. Envisionment	Models of how people reason about causality, possibility, and intervention in real and imagined worlds
6. Explanation	Models of the process of generating explanations for events and states with unknown causes
7. Expectation	Models of people come to expect that certain events and states will occur in the future, and how they handle expectation violations
8. Theory of Mind reasoning	Models of how people reason about the mental states and processes of other people and themselves
9. Threat detection	Models of how people identify threats and opportunities that may impact the achievement of their goals
10. Goal management	Models of how people prioritize and reconsider the goals that they choose to pursue
11. Planning	Models of the process of selecting a course of action that will achieve one's goals
12. Design	Models of how people develop plans for the creation or configuration of an artifact, process or information
13. Scheduling	Models of how people reason about time and select when they will do the plans that they intend to do
14. Decision making	Models of how people identify choices and make decisions
15. Monitoring	Models of how people divide their attention in ways that enable them to wait for, check for, and react to events in the world and in their minds
16. Plan execution	Models of the way that people put their plans into action and control their own behavior

Not all the aspects mentioned by Gordon are required for our task, and neither are they for many other tasks for which cognitive agents have been developed. In contrast with Gordon who started his listing of cognitive capabilities from a natural science, namely psychology, Langley et al. (2006) present a list of cognitive capabilities taking a design science approach. Langley et al. discuss the capabilities and functionalities

that a cognitive agent could embed by studying architectures developed for the goal of intelligent behavior generation. They divide these capabilities into nine main areas, see Table 2.2. For clarity of presentation, their last capability 'Remembering, Reflection and Learning' is split into the two capabilities 'Remembering and Reflection' and 'Learning'.

Table 2.2: The 9 Capabilities of Cognitive Agents according to Langley et al. (2006)

Capability	**Summary**
1. Recognition and Categorization	The ability to recognize situations or events as instances of known or familiar patterns
2. Decision Making and Choice	The ability to make decisions and select among alternatives
3. Perception and Situation Assessment	The ability to sense, perceive, and interpret some external environment
4. Prediction and Monitoring	The ability to predict future situations and events accurately
5. Problem Solving and Planning	The ability to generate plans and solve problems
6. Reasoning and Belief Maintenance	The ability that lets an agent augment its knowledge state
7. Execution and Action	The ability to execute skills and actions in the environment
8. Interaction and Communication	The ability to communicate and transfer knowledge from one agent to the other
9.a. Remembering and Reflection	The ability to encode and store the results of cognitive processing in memory and to retrieve or access them later
9.b. Learning	The ability to generalize beyond specific beliefs and events

We appreciate the pragmatics of the work of Langley et al. compared to that of Gordon, and decided to take their capability classification as the basis for our Capability Description Framework, see Chapter 6. Even more pragmatic is the division followed by Pew and Mavor (1998), who discuss architectures for modeling individual human behavior with the specific focus of their application to military simulations. Because of its military focus, this piece of work functioned as one of the starting points of our research. Pew and Mavor analyze existing architectures on six key areas, see Table 2.3.

The first five areas can be considered cognitive capabilities; the latter however is an odd one out. Nevertheless, behavior moderators are important for modeling cognition and human behavior, since human behavior is known to vary due to internal aspects as

Table 2.3: The key areas of architectures according to Pew and Mavor (1998)

Key Area
1. Attention and Multi-Tasking
2. Memory and Learning
3. Planning
4. Decision Making
5. Situation Awareness
6. Behavior Moderators

stress and emotion. Behavior moderators enable the modeling of this variety of behavior, making it more human-like. In the subsequent sections we elaborate on such behavior moderators in respect to the modeling of human suboptimal task performance, with a specific focus on the modeling of cognitive biases.

This section illustrates that there is no clear consensus on which capabilities constitute cognition. Cognitive Science is occupied with clarifying this fundamental question; Artificial Intelligence on the other hand simply models those aspects that are sufficient for generating intelligent, human-like behavior for a given task. For our purpose it holds that the cognitive capabilities that need to be modeled will vary between tasks and training objectives, e.g., an opposing naval force does not need to be able to communicate.

2.2.2 Cognitive Biases

Previously, we explained that the modeling of cognitive biases is relevant for the training of military personnel. On the one hand it is important to train them in recognizing and dealing with their own biases, on the other hand they need to be trained on dealing with the biased behavior of their team mates.

This section elaborates on research concerning cognitive biases. It starts with several examples of cognitive biases, after which the general limitations of human cognition are discussed that by themselves might lead to suboptimal behavior. Although the previous section only mentioned capabilities as the parts that make up cognition, cognitive processes are in addition influenced by other processes, e.g., emotion and stress. At the end of this section we elaborate on the role these so-called behavior or performance moderators play in the functioning of cognition.

When the cognitive mechanisms underlying biases are known, it will be possible to formalize these mechanisms in a cognitive agent model. This model can then be embedded in a software agent that will therefore be able to display more human-like, possibly biased, behavior. In addition it can be used to compare the behavior of the trainee with, so that possible occurring biases can be detected.

Cognitive Bias Examples

Dozens of cognitive biases, operating on various levels and influencing a variety of cognitive processes, have been identified. For extensive discussions on human biases see Baron (2000), Reason (1990), who refers to them as error forms, or Pohl (2004), who calls them cognitive illusions. To facilitate the discussion of cognitive biases we divide them in 4 classes that differ in the level at which they operate: social (attributional), memory, judgment, and decision-making biases. This is no clear-cut division, many biases or their underlying mechanisms overlap categories, e.g., a mechanism that leads to a judgment bias might a the same time lead to a decision-making bias.

A *social or attributional bias* is a cognitive bias that affects the way it is determined who or what was responsible for an event or action (attribution). A well-known example is the *correspondence bias*, which is 'the tendency to draw inferences about a person's unique and enduring dispositions from behaviors that can be entirely explained by the situations in which they occur' (Gilbert and Malone, 1995). In other words, people have an unjustified tendency to assume that a person's actions are not the result of that person's social and environmental situation, but of the 'kind' of person that person is.

A *memory bias* is a cognitive bias that either enhances or impairs the recall of a memory, or that alters the content of memory that is claimed to be recalled. Many memory biases are simply referred to as effects. Two well-known memory effects are primacy and recency: the fact that respectively the first and the last items on a list show an advantage in memory, i.e., are easier recalled than the items in the middle of the list. An example of a memory bias is the *hindsight bias*, which is 'the tendency for people with outcome knowledge to believe falsely that they would have predicted the reported outcome of an event' (Hawkins and Hastie, 1990). In other words, people have an unjustified tendency to claim after events have happened that they 'knew it all along'.

A *judgment bias* is a cognitive bias that affects estimated probabilities or beliefs, and can therefore have impact on the quality of human performance. Two examples of judgment biases are also referred to as primacy and recency, but are now defined as the tendency to weigh initial events more than subsequent events, and the tendency to weigh recent events more than earlier events, respectively. Multiple researchers (Adelman et al., 1996; Wang et al., 2000) have shown that these effects occur in tactical decision-making. For example, when participants were presented the same friendly/neutral/hostile evidence concerning yet unidentified tracks, their order influenced the participants' final judgment concerning the contact's identity. Another well-know judgment bias is the conjunction fallacy, which is the tendency to 'regard a conjunctive event as more probable than one of its components' (Tversky and Kahneman, 1982). For example, Tversky and Kahneman (1982) show that people judge the probability of 'Linda is a bank teller

and is active in the feminist movement' as being higher than the probability of 'Linda is a bank teller'. Several other judgment biases will be discussed in the next section.

A *decision-making bias* is a cognitive bias that affects the way a decision is made. This can happen in a multitude of ways, among others by a false probability estimation, i.e., through a judgment bias. A wide-spread decision-making bias is the confirmation bias, which is the tendency to search for or interpret new information in a way that confirms one's preconceptions and opinions, and to ignore, undervalue, or not look for information which contradicts these prior beliefs. This bias has been shown to occur frequently in military tactical decision-making (Fewell and Hazen, 2005).

Mechanisms underlying Cognitive Biases

The study of cognitive biases progressed greatly by the work of Tversky and Kahneman (1974) who investigated in laboratory settings human judgment and decision-making. They compared the judgments and decisions actually made by humans with those that should be made if they would follow the rules of rational choice theory. Rational choice theory assumes that humans select the best option given the current circumstances, and that that value judgment is determined by a stable preference function. However, Tversky and Kahneman found that human choices differ in systematic and predictable ways from the rational choice. These ways are referred to as cognitive biases.

Twenty years earlier Simon (1956) was among the first to oppose the view of humans as rational entities that can calculate the best option; a view attributing humans with unlimited computational powers and perfect knowledge. Norman and Bobrow (1975) stress that human processes are limited exactly by these two factors: a process can be limited in its performance by limits in the amount of available processing resources (such as memory or processing effort) or by limits in the quality of the data available to it. In his work Simon proposes that humans are rational but under the constraints of limited time, knowledge and computational capacities. He states that humans should not aim for an optimal solution, but search until a solution is found that is *satisficing*.

It is often claimed that the origin of cognitive biases lies in the limitations of human information-processing capacity. It is generally acknowledged that human cognition is limited in the amount of information it can attend to and process at a single time. The classical paper by Miller (1956) stresses that the capacity of short-term memory for chunks of information is limited to 'the magical number seven, plus or minus two'. Others have researched and identified limits in cognitive processing capacity, which influences the ability to perform multiple tasks simultaneously (e.g., Kahnemann, 1973). The limited capacity of human cognition gives rise to specific processes, which are, or might lead to, cognitive biases.

Heuristics Tversky and Kahneman (1974) claim that the biases they identified are, at least partly, the result of decision-making using heuristics. Heuristics, also referred to as rules of thumb, are mental shortcuts that humans use to speed up and simplify their decision-making process. It is assumed that heuristics have evolved due to, and are applied in reaction to, the limitations of cognition. Tversky and Kahneman identify three such heuristics:

- **Representativeness** - the tendency to evaluate the probability of object or event A to belong to, or originate from, class or process B by the degree to which A resembles, or is representative for B.
- **Availability** - the tendency to assess the frequency of a class or the probability of an event by the ease with which instances or occurrences can be brought to mind.
- **Adjustment and anchoring** - the tendency to make an estimate by starting from an initial value and adjusting that to yield the final answer, which unfortunately leads to an answer that is biased toward the initial value.

Heuristics speed up the decision-making process and often succeed, but sometimes fail. The Kahneman and Tversky's heuristics-and-biases program (Kahnemann et al., 1982) focused on these failures by constructing decision problems that led to structural decision errors (cognitive biases) caused by the use of heuristics. As a result heuristics received a negative connotation.

Gigerenzer and colleagues (Gigerenzer et al., 1999) oppose this view and stress above all the usefulness of heuristics. Their research explores 'fast and frugal heuristics - simple rules in the mind's adaptive toolbox for making decisions with realistic mental resources'. Todd and Gigerenzer (2000) state that heuristics can enable both living organisms and artificial systems to make smart choices quickly and with a minimum of information by exploiting the way that information is structured in particular environments. The latter aspect is also referred to as a heuristic's ecological rationality; the degree to which it is adapted to the structure of an environment.

Hertwig and Todd (2003) go even further in questioning the negative status of cognitive limitations and forward the thesis that they actually enable, instead of disable, important adaptive functions. Along the same line, Schooler and Hertwig (2005) introduce a model that suggests that forgetting facilitates human inference performance by strengthening the chain of correlations, linking the target criteria, environmental frequencies, and fundamental memory-retrieval processes. Still, the argument that heuristics are adaptive given their ecological rationality implies that when they are applied in a differently structured environment, they might lead to wrong behavior.

Error Phenotype versus Error Genotype A useful distinction when talking about biases is made by Hollnagel (1993). Hollnagel introduces the notion of error phenotype for the observable manifestation of an error, and the notion of error genotype as the mental activity that supposedly underlies and produces that observable manifestation. Many cognitive biases are the phenotypes of heuristics. Tversky and Kahneman (1974) describe the three heuristics mentioned above, but the number of biases they describe that these heuristics lead to is significant larger. One example of a cognitive bias following from a heuristic is the previously described conjunction fallacy. Tversky and Kahneman (1982) note about this:

> "A conjunction can be more representative than one of its constituents, and instances of a specific category can be easier to imagine or to retrieve than instances of a more inclusive category. The representativeness and availability heuristics therefore can make a conjunction appear more probable than one of its constituents."

However, by no means does the application of a heuristic lead in all situations to the emergence of a bias. On the other hand, for memory biases and effects it is generally the case that they are the observed 'error' phenotypes from the 'error' genotypes that are the basic mechanisms of memory. Therefore, memory biases and effects are a constant finding within and between subjects.

Cognitive Biases in Real Life Most studies into cognitive biases have taken place in laboratory settings. However, they do occur in real life, and influence human judgment and decision-making: various famous accidents in military warfare have been attributed to false judgments or decision-making under influence of cognitive biases. Perrin et al. (1993) examine in an empirical study human judgment biases under conditions of uncertainty and time pressure in surface Anti-Air Warfare (AAW). To be precise, they studied whether the judgments of naval tactical action officers in a realistic task simulation exhibit characteristics of the heuristics and biases of availability, representativeness, anchoring-contrast, and confirmation. This is what they found:

> "Our subjects ignored baseline trends when other case-specific information was available (representativeness and availability). They were significantly influenced by the order they received evidence, showing a recency effect characteristic of contrast. Additionally, as is characteristic of confirmation bias, they recalled much more of the information that was consistent with their final hypothesis and evaluated it as more informative than the inconsistent data, regardless of which hypothesis they had adopted."

Manifestations of Cognitive Biases

The manifestation of cognitive biases can be discussed from two viewpoints. First, its timing, a dynamical aspect, can be discussed. In other words; it can be investigated which external and internal circumstances might inflict the occurrence of a cognitive bias. Second, its positioning, a static aspect, can be investigated: which part of the cognitive process it influences.

Positioning of Biases When we divided cognitive biases in several classes in Section 2.2.2, we touched upon the issue of their position, namely as influencing social, memory, judging or decision-making processes. Another way to position them is by means of the Skills, Rules, Knowledge (SRK) framework developed by Rasmussen (1983) to define three types of cognitive processes present in operator information processing. Reason (1990) classifies human errors by means of these three process types as follows:

- **Skill-based level:** At the skill-based level, human performance is governed by stored patterns of preprogrammed instructions represented as analogue structures in a time-space domain. Errors at this level are related to the intrinsic variability of force, space, or time coordination.
- **Rule-based level:** The rule-based level is applicable to tackling familiar problems in which solutions are governed by stored rules (productions) of the type if (state) then (diagnosis), or if (state) then (remedial action). Here, errors are typically associated with the misclassification of situations leading to the application of the wrong rule, or with the incorrect recall of procedures.
- **Knowledge-based level:** The knowledge-based level comes into play in novel situations for which actions must be planned on-line, using conscious analytical processes and stored knowledge. Errors at this level arise from resource limitations ('bounded rationality') and incomplete or incorrect knowledge. With increasing expertise, the primary focus of control moves from the knowledge-based towards the skill-based levels; but all three levels can co-exist at any one time.

Another classification that can be made is on the level at which cognitive biases have impact: *within* a process, or *between* processes. For example, most memory biases influence the recollection *process* of memories, i.e., which memory is retrieved given a certain query. Other biases might influence *which* recollection process executes, i.e., which query is executed. The confirmation bias is a good example to clarify this distinction further, as it operates within and between cognitive processes. Within a specific cognitive process the confirmation bias influences the weight given to (dis)confirming

information. On the process level it influences the next information-searching action in the world, and biases this to confirming instead of conflicting information.

Although it might be useful to train military personnel in recognizing and dealing with all types of cognitive biases, we focus on a subset that is relevant to the cognitive models we develop. In Chapters 3 and 4 we focus on the development of a belief and memory framework for cognitive agents, and model biases that occur within processes, like in the retrieving and reasoning upon information. In Chapter 5 we focus on the development of two control frameworks for cognitive agents, for which we model biases and heuristics that occur between processes, so on the level of control.

Timing of Biases We have not elaborated extensively on *when* heuristics and biases occur. We do have mentioned that they are generally considered to be a consequence of the limitations of human cognition. When there is too much information to take into account, or too little time to do so, humans apply heuristics and biases might occur.

A second generally held opinion is that they occur especially under conditions of stress and fatigue, e.g., Cohen et al. (1986) mention that everybody finds that slips of actions increase with tiredness and stress. The general idea behind this is that these circumstances shrink the cognitive capacity of humans, and make the limitations of the cognitive system more prevalent. For example, the results of a study by Harris et al. (2005) on the influence of extended stress on human performance support the model that stress decreases resource reserves. They state that the extended stress circumstances decrease the participants' ability to continue to mobilize the resources required to perform complex tasks. For a more detailed account of the influence of stress, workload, and fatigue on cognition, see, e.g., Reason (1988) and Hancock and Desmond (2001).

Studying the underlying mechanisms of heuristics and cognitive biases is useful, not only to understand them, but also to model them. In the next section we introduce various approaches toward the modeling of cognition, as well as integrated architectures, which are implemented theories of the mechanisms underlying human behavior. In addition, we elaborate on the modeling and appearance of cognitive biases in two of these integrated architectures.

2.3 Models of Cognition

The previous section elaborated on the fact that cognition comprises a wide variety of cognitive capabilities. In addition, it discussed cognitive biases and how these, to a smaller or larger extent, influence these capabilities. Dividing cognition in clear conceptual parts, like capabilities, facilitates the modeling of such parts. In particular, because

each of these parts can be modeled by a modeling approach most suited. However, the definition and modeling of 'parts of cognition' is not enough for implementing a cognitive agent model. In addition, it has to be determined how these parts interact and in which order they operate. In the introduction of Chapter 5, Control Framework, we will elaborate on this topic.

We start this section with a discussion of a variety of modeling approaches that have been used to model (parts of) cognition. Next, we elaborate on integrated architectures that stress the importance of studying the integrated architecture underlying cognition, instead of only focusing on the modeling of its parts.

2.3.1 Modeling Approaches

Nowadays, most researchers agree that hybrid modeling is the appropriate way to model cognition. Hybrid modeling is used as a term to express the fact that a cognitive model embeds qualitative (symbolic) as well as quantitative (numeric) elements. In the models developed in this study both qualitative and quantitative elements play a role; the former to clearly label knowledge, the latter to handle this knowledge in subtle ways.

Hybrid models acknowledge the intuitive feeling that humans embed two approaches toward the processing of cognitive tasks (Smolensky, 1988; Sloman, 1996). First, they can reason consciously following certain rules and algorithms, and these can be captured well by qualitative systems. But second, humans execute cognitive processes on a more unconscious, automatic level formed by associations, which are better captured by quantitative means. Qualitative systems are well-suited to generate rational behavior in certain tasks, but they usually lack the means to generate typical aspects of human behavior, like the ability to cope with uncertain or incomplete information. In order to model human behavior in all its aspects, a hybrid modeling approach is required (Minsky, 1992). Sun (2002b), and more recently Bader and Hitzler (2005), provide comprehensive surveys of various types of hybrid systems.

Qualitative Modeling Approaches

A few decades ago it was thought that 'a physical symbol system has the necessary and sufficient means of general intelligent action', as expressed by the Physical Symbol System Hypothesis of Newell and Simon (1976). This view gave rise to many qualitative systems of cognition that solely consist of symbols and mechanisms that operate on these symbols independent of their content.

In general, a qualitative system capable of displaying cognitive behavior embeds a symbolic model of its environment and a symbolic specification of actions that it can

perform. In addition, it should embed an inference mechanism. Qualitative systems frequently use condition-action rules, in which case they are referred to as rule-based systems, or a logic engine. Other examples of qualitative modeling approaches are causal networks whose connections denote causal relations in a binary fashion, and semantic networks, in which the connections between knowledge elements are labeled with meaningful symbols, e.g., isa, or has.

A specific subset of systems embedding qualitative elements are formed by systems whose symbols are labeled with the mentalistic notions *belief*, *desire* and *intention*. The idea that these three concepts suffice to generate intelligent action dates back to Aristotle (B.C. 350) and was revived by the work of Bratman (1987) on rational agents. For so-called BDI agents it holds, globally stated, that beliefs represent the agent's knowledge (its information state), while desires represent its goals (the agent's motivational state). What intentions exactly represent varies between approaches, but in general the agent's deliberation state. In the formal BDI-framework provided by Rao and Georgeff (1991) intentions are a separate notion which cannot be reduced to the concepts of beliefs and goals (desires), while Cohen and Levesque (1990) only take beliefs and goals as primitives and consider intentions to be a subset of the goals. The main stream of agent modelers uses intentions to express the agent's commitments to certain plans that will lead to achievement of the goals formed by its desires.

Using mentalist notions in agent systems has several advantages. First, the specific representation of beliefs and goals facilitates autonomous goal-directed behavior tailored to the current state of the world. Second, agents that are based on mentalist notions are intuitively understandable. Notions like belief, desire, and intention stem from folk psychology: they map easily to the language people use to describe their reasoning and actions in everyday conversation (Norling, 2004). For a panel discussion on the BDI model of agency, in specific on how it stands in relation to other qualitative models of agency and on its limitations to capture certain (human-like) types of behaviors, see Georgeff et al. (1999).

Quantitative Modeling Approaches

Previously, we stressed the benefit of hybrid cognitive models because although qualitative models are well-suited to model specific aspects of human behavior, they are less suited for others. In this section we elaborate on a subset of the quantitative models that have been proposed as alternatives for the qualitative models of cognition. For extensive discussions on many other mechanisms to implement reasoning on quantitative data, e.g., fuzzy logic, frequency probability, and utility theory, see Russell and Norvig (2003).

Connectionist Systems Connectionism is inspired by the structure of the brain, and views cognition as the emergence of global states in a network of simple components. Connectionism tries to model cognitive behaviors by interconnecting many simple processing elements, referred to as nodes, that can have a small local memory and form together with their connections a so-called Artificial Neural Network (ANN). When the network is active its nodes operate in parallel on their local data and the numerical (instead of symbolic) information that they receive through their connections. Within a connectionist model knowledge is distributed and resides in the weights of the connections between the simple units.

Various types of artificial neural networks can be distinguished based on their overall structure and way in which they are formed, see e.g. Maes (1994) and Browne and Sun (2001). One important aspect that ANNs can vary in is on whether they incorporate distributed or local representations. Local representation networks resemble the qualitative approach in that they use single nodes to represent symbols. However, the rules operating on these symbols are not represented symbolically like in causal or semantic networks, but formed by the numerical properties of the nodes and connections: activation thresholds and strengths respectively. Neural networks incorporating distributed representations do not capture meaning in one single node, but let it emerge from the interaction of multiple neurons. This makes those neural networks more biological plausible, but also more complex and harder to analyze than the local representation versions. Another drawback of such neural networks is that for forming ('training') them a large amount of training data, generally in the form of thousands of examples, is required.

In general, connectionist systems are formed by training them on a large amount of data using a variety of techniques (e.g., supervised, unsupervised, or reinforcement learning). However, this data might not always be available. An alternative approach to ANNs to deal with the lack or ambiguity of information as often encountered in the real world are Bayesian systems.

Bayesian systems A Bayesian system consists of a network whose nodes represent (any kind of) variables, and whose connections encode the causal dependencies among the variables; missing connections encode causal independencies between variables.The dependencies are quantified by conditional probabilities for each node given its parents in the network. Bayesian networks can be used to model a large number of cognitive processes: reasoning (using the Bayesian inference algorithm), learning (using the expectation maximization algorithm), planning (using decision networks) and perception (using dynamic Bayesian networks), see Russell and Norvig (2003).

Dynamical Systems Although connectionist and Bayesian systems are more biologically plausible than symbolic systems, they still comply to the view of cognition as an information processing system that converts a set of inputs into an set of outputs. In contrast, dynamical systems view cognition as a situated activity and model it as part of a system that also encompasses an agent's body and world (Beer, 2000).

Dynamical system theory states that 'Natural cognitive systems are certain kinds of dynamical systems, and are best understood from the perspective of dynamics' (Gelder, 1995). Dynamical Systems consider cognition to be a multiple-dimension space of thoughts and behaviors that is explored when thinking. For modeling cognition as a dynamical system it has to be described how this space of thoughts and behavior is explored under influence of external and internal pressures. This is usually done by differential equations. Knowledge is distributed within a dynamical system, namely over many different kinds of processes, each represented by a numerical equation.

Selected Modeling Approach

Most of the research on modeling synthetic entities for training simulations embed an approach that is mainly qualitative in nature, see Section 2.4.1. Reason for this is that qualitative models offer conceptual and practical benefits. First, 'mental states' are transparent using this approach, and it is easy to backtrack which part of the model is responsible for a certain behavior. Second, because the model is made up of discrete parts, it is easy to access and alter one part of the model without affecting the rest. Third, discrete parts facilitate reuse of parts of the model. However, qualitative models are not well suited for modeling some fundamental characteristics of cognition, like the ability to learn and to deal with uncertain or incomplete information[1]. To model such aspects, frequently quantitative elements are embedded within qualitative models. This endangers the benefits of the latter: its functioning becomes less transparent, and the knowledge acquired during execution and stored in quantitative terms is difficult to reuse.

Sloman (1997) argues that selecting the 'best' approach for modeling cognition for a specific task is a matter of analyzing trade-offs. We know that our goal is the development of cognitive models that enable agents to fulfill tasks within simulator training of military tasks currently fulfilled by humans. In Section 1.1.5, we denoted several requirements for these agents. Among others, they need to reduce costs of training, which makes it undesired that a new agent has to be developed from scratch for every scenario. Furthermore, training requires that the behavior of such agents is varied but tunable,

[1] Qualitative approaches towards these aspects do exist. For example, inductive logic programming is a type of machine-learning for logic programs, and uncertainty can qualitatively be denoted by, e.g., comparative probability, linguistic labels, or partial preference orderings (Parsons, 2001).

which requires a clear understanding of the source of the behavior. These requirements motivate us to model cognition based on qualitative representations. Another reason for this is that we want to base our models on the mentalistic notions of beliefs and goals, because of the benefits that mentalist notions bring to the modeling and understanding of agent behavior. However, we are also interested in the modeling of more human-like behavior, like the ability to deal with information with varying degrees of certainty, and the fact that humans almost never behave in the same way. We want to model variety in behavior due to the occurrence of cognitive biases that can influence cognitive processes to a smaller or larger extent. These aspects are hard to model following an approach that is purely qualitative in nature. Therefore, we also embed quantitative aspects in our models.

As an outlook: in Chapter 3 we attach a quantitative certainty value to beliefs of agents, and reason about the classification of a radar contact using a naive Bayesian classifier; in Chapter 4 we determine and use availability values of beliefs; and in Chapter 5 we reason about the relevancy (utility) of reasoning components, among others based on the agent's cognitive exhaustion level, a quantitative value.

2.3.2 Integrated Architectures

In the introduction chapter we quoted Lewis (2001) who states that 'cognitive architectures constitute a fixed set of processes, memories and control structures that define its underlying theory about human cognition'. This definition of cognitive architectures supports our previous remark that for implementing a cognitive agent model not only 'parts of cognition' need to be modeled, but also the way in which these interact, i.e., their control. Newell (1990) was among the first to stress the importance of studying the integrated architecture underlying cognition. He phrased this as the need for a unified theory of cognition: a single set of mechanisms that together account for all aspects of cognition, similar to the mind. A few years later Sloman (1997) expresses the same need, i.e., the need to study architectures to research how diverse but interrelated capabilities can be put together.

Fulfilling their call, many integrated architectures have been developed in the last two decades that facilitate the building of software agents. An agent is formed when task-specific declarative and procedural knowledge is added to the architecture that takes care of everything else, like the functioning of the agent's memory and how it decides on its actions. How this is done greatly varies between architectures. The work of Pew and Mavor (1998), later extended by Ritter et al. (2003), provides the most extensive and comprehensive overview and comparison of existing architectures for modeling human behavior to date. For another overview, see Morrison (2003).

This section is about architectures, to investigate whether existing architectures can be used to build software agents for our research objectives. We start with a general discussion on the suitability of various architectures to model cognition and human behavior. To increase the readability of that section, we do not elaborate on the architectures and software packages mentioned. Instead, Appendix A lists and shortly discusses the various architectures and languages referred to throughout this dissertation. Next, we elaborate in more detail on two cognitive architectures frequently used for the development of cognitive agents, Soar (Laird et al., 1987) and ACT-R (Anderson and Lebiere, 1998; Anderson et al., 2004). We discuss their applicability in general, as well as their relevance for our research objective. Hereby specific attention is paid to their ability to model cognitive biases.

Suitability of Architectures

In the beginning of this study we investigated several integrated architectures to gain insight in which cognitive architectures would be (most) relevant to the research questions posed. This turned out to be a hard question, because architectures are very diverse and comparing them is like comparing apples to pears. Simon (1997) notes about existing integrated architectures that they can be seen to be built around a core cognitive activity, which is then extended to handle other cognitive tasks, e.g., the core cognitive activity of ACT-R is semantic memory, in Soar the core is problem-solving.

From reviews on the modeling of human behavior for military simulations (Pew and Mavor, 1998; Banks and Stytz, 2003) it follows that there exists no consensus on the way, or architecture, that is best suited for this task. Van Lent et al. (2004) subscribe to this, as in their research three separate human behavior models are used to support three different roles for synthetic entities in a single simulated scenario. They select for each role an architecture suited to model the requirements posed for each role:

> "The Soar architecture, focusing on knowledge-based, goal-directed behavior, supports a fire team of U.S. Army Rangers. PMFServ, focusing on a physiologically/stress constrained model of decision-making based on emotional utility, supports civilians that may become hostile. Finally, AI.Implant, focusing on individual and crowd navigation, supports a small group of opposing militia."

Morgan et al. (2005) try to get insight into the strengths and weaknesses of architectures by the introduction of dTank. dTank is a competitive environment which can be used for architectural comparisons of competitive agents, and for comparisons of human and agent behavior. They used dTank to compare models built in Java, Jess and Soar. The development of dTank is continued (Ritter et al., 2007a) and additional models are developed, most recently a CoJACK model (Evertsz et al., 2008b). Morgan et al. (2005)

state that only few modelers have used the same environment to examine the behavior of humans as well as that of agents built in different architectures.

A noticeable exception is the Agent-based Modeling and Behavior Representation project (AMBR), initiated in 1999 by the Air Force Research laboratory (AFRL). AMBR focused on comparing models of complex human behavior. Gluck and Pew (2005) cover its methods and results. In short, four research groups were asked to develop models for the same two tasks. Each group used a different architecture, which resulted in a comparison between ACT-R, COGNET/iGEN, D-COG, and EPIC-Soar. The focus of the first experiment was modeling multi-tasking; the second was modeling category learning. The domain for both experiments was a simplified version of Air Traffic Control.

The conclusion of this project was that 'the quality/acceptability/appropriateness of a model, and any effort to rank order it relative to models developed with other architectures, depends on what one values in a model and an architecture.' Gluck and Pew mention several factors on which models can be evaluated and compared: goodness of fit of the model data with the human data; the degrees of freedom available in implementing the model; how much of the model was reused from previously implemented models; the interpretability of the model's behavior during run-time; and the generalization of the model. These factors are likely to vary between the implemented models, between the architectures used for this implementation, and even more general between the various approaches toward the modeling of cognition implemented in these architectures.

Therefore, it is difficult to select an architecture which is best suited for modeling cognition and thus human behavior. Fortunately, our goal is not to model cognition in all its facets, but to develop cognitive models for tactical training simulations that will enable software agents to play a role in a human-like way, and that aid in providing cognitive feedback. Model requirements for these objectives are listed in Section 1.1.5, which made us decide to model cognition using mentalistic notions and following a hybrid approach (Section 2.3.1). Below we introduce the cognitive architectures Soar and ACT-R, and discuss their applicability and relevancy for our research objectives.

Soar

Description The Soar architecture implements a qualitative approach toward the modeling of cognition (Laird et al., 1987), and is proposed by Newell (1990) as 'a candidate unified theory'.

Soar's basic assumption is that human behavior can be modeled as a set of parallel operating and interacting processors, to be precise by a cognitive, perceptual and motor processor. Soar's perceptual and motor processors provide the means of perceiving and acting upon an external world. In addition, Soar incorporates an unbounded work-

```
Soar
  while (HALT not true) Cycle;

Cycle
  InputPhase;
  ProposalPhase;
  DecisionPhase;
  ApplicationPhase;
  OutputPhase;


ProposalPhase
  while (some I-supported productions are waiting to fire or retract)
    FireNewlyMatchedProductions;
    RetractNewlyUnmatchedProductions;

DecisionPhase
  for (each state in the stack,
       starting with the top-level state)
  until (a new decision is reached)
    EvaluateOperatorPreferences; /* for the state being considered */
    if (one operator preferred after preference evaluation)
      SelectNewOperator;
    else                    /* could be no operator available or */
      CreateNewSubstate;    /* unable to decide between more than one */

ApplicationPhase
  while (some productions are waiting to fire or retract)
    FireNewlyMatchedProductions;
    RetractNewlyUnmatchedProductions;
```

Figure 2.1: A simplified version of the Soar Algorithm (Laird et al., 2006))

ing memory through which the three processors communicate. This working memory symbolically represents the current state of knowledge, which comprises the inputs of perception, the output parameters for the motor modules, and purely internal structure. These symbolic representations are called *working-memory elements*.

Soar assumes human behavior to be goal-oriented. It implements this as a search through interacting problem spaces, where each problem space contains a set of symbolic production rules called *operators* that are applied to states to produce new states. While searching, the model learns the results of its problem solving. A task, or goal, is modeled by the specification of an initial state and one or more desired states.

When the model is run, symbolic production rules called *productions* match the contents of working memory and can either directly retrieve additional information from long-term memory (automatic association; an unconscious reasoning step) or propose to apply a certain operator (embedding a conscious reasoning step). These productions match and fire in parallel, and therefore multiple operators might be proposed at the same

time. However, only one operator can execute each cycle, but the rules embedded in an operator can fire simultaneously. Which operator is applied is determined by weighing the operators' *preferences* (qualitative labels denoting its priority), which can be added to operators by productions.

When sufficient knowledge is available in the problem space for a single operator to be selected and applied to the current state, the behavior of a Soar model is directed and smooth, as in skilled human behavior. Whenever knowledge is lacking and the active preferences are not sufficient to allow a unique choice, the architecture responds by setting a subtask to resolve this state called *impasse*, and the entire process recurs. For a simplified version of the Soar algorithm, see Figure 2.1.

If the recursive processing adds new preferences to operators that are active in the original task, Soar's learning mechanism can create a new association between those working-memory structures in the original task that led, through a chain of associations in the subtask, to the new preferences in the original task. This learning mechanism is called *chunking* and can be turned on or off.

The initial knowledge of a Soar model has to be encoded by the human modeler in symbolic production rules. The initial state of the model is often coded manually, but a model can also start with no knowledge of the problem state and just acquire that knowledge from the external environment.

Applicability Soar has been evaluated for its suitability to model human behavior for a wide variety of tasks, among others military. It is frequently used to develop agents that act as (large) expert systems. For this, knowledge off-line encoded in Soar is used on-line by the agent to decide on its actions and to solve problems.

A well-known example of such an effort is TacAir-Soar (Jones et al., 1999), which is a fully autonomous Soar system that embeds behaviors from the tactical air domain. TacAir-Soar's knowledge base consists of over 7500 rules, which provide agents with behaviors for a wide range of simulated aircraft and missions. The agents, once briefed, can autonomously plan and execute their missions using US doctrine and tactics. TacAir-Soar can be used to model friendly, opponent, as neutral forces in simulated environments.

Soar has also been used to model opponents (Wray et al., 2002, 2004) and subordinate team members (Van Lent et al., 2004) as part of the Virtual Training & Environments (VIRTE) program of the American Office of Naval Research (ONR). This program focuses on developing virtual trainers for Military Operations on Urbanized Terrain (MOUT). For this program synthetic entities were modeled as either opponents or members of a fire team consisting of four U.S. Marines. This fire team is situated in a virtual urban environment and has the task to clear a building that possibly contains

enemy soldiers. The environment and synthetic entities were developed using the Unreal Tournament game environment. Wray et al. (2002) list five high level requirements for the intelligent opponents of the fire team in the domain described: Observational fidelity; Faster than real-time performance; Behavior variability; Transparency; and Rapid, low-cost development. Later on Wray et al. (2004) extend this list with the requirements of 'Competence' and 'Taskability', and transform 'Faster than real-time performance' to 'Minimal computational footprint'. Most of these requirements have also been listed in Section 1.1.5 as required for the cognitive software agents that are to play a role within a simulated training environment. Wray et al. (2004) conclude their paper with the statement: 'Achieving intelligent opponents that are fully autonomous, believable, and interactive, while exhibiting a variety of behaviors in similar circumstances, will require much additional research and development.' More specific, they mention that 'less' competent behaviors (possibly representing poorly trained guerrilla's or novices), and additional errors in judgment and perception could be modeled.

The team members modeled by Van Lent et al. (2004) are of less interest to our research objectives than the opponents, since these team members were not autonomous; a human player directed their mission-specific behavior by issuing specific commands.

Relevancy The Soar architecture is interesting for our goal of modeling cognitive agents for training simulations, since it is shown to be sufficiently powerful to generate believable individual human behavior in military situations. In addition, it is sufficiently robust to run large-scale models connected to real-world systems in real time. However, most Soar models are developed to display only one type of behavior, namely expert behavior.

Randolph M. Jones (SoarTech, private conversation at ICCM 2007) explained that it is hard to have the TacAir-Soar agents display wrong behavior. In one project the goal was to train oil rig operators in handling pilots that fly in too steep. This training should take place in a simulated environment with TacAir-Soar agents controlling these pilots. Because TacAir-Soar agents normally display correct behavior, their procedure on approaching a platform had to be altered so they would fly in too steep. Unfortunately, this did not work, because as soon as the behavior was obviously false, e.g., they would fly too quick for their height, other behavior rules would take over and correct the behavior.

So, although the Soar architecture looks promising for generating human behavior, it remains to be seen how suited it is for displaying behavior at a variety of performance levels. One reason why this may be hard is that Soar does not embed the cognitive limitations found in humans, which possibly give rise to cognitive biases. Most obviously, it embeds an unbounded working memory while it is uncontentious that human working memory is limited.

Interestingly, research has been done on embedding performance moderators in Soar. Chong (1999) models fear in Soar. Unfortunately, this is not a dynamical model and thus does not allow for changes in this moderator value: the models start and stay fearful. Gratch and Marsella (2001) extend Soar with a model of emotional appraisal called Émile. Émile embeds mechanisms for evaluating how environmental events relate to an agent's plans and goals, and how these evaluations give rise to emotions. However, the effects of these emotions on behaviors are left to be determined by the modeler. In more recent work, Marinier and Laird (2007) incorporated a computational model of emotion, mood, and feeling in Soar, and demonstrate that such a model can help direct problem solving, and speeds reinforcement learning.

Although all these models are interesting for the purpose of developing software agents that show more human-like behavior, none of them is tailored to model the occurrence of cognitive biases. The modeling of more human-like behavior due to the incorporation of emotions is not one of the topics of this dissertation. This study attempts to model more human-like behavior by incorporating cognitive models that show varied behavior, with (part of) this variety stemming from the occurrence of biases.

ACT-R

Description The ACT-R architecture implements a hybrid approach toward the modeling of cognition (Anderson and Lebiere, 1998). ACT-R constitutes a unified theory of cognition and is based on detailed findings concerning the functioning of human memory and of learning and problem-solving processes.

ACT-R incorporates a declarative and a procedural memory module, as well as a perceptual (visual) and motor (manual) module that provide the means for perceiving and interacting with an external world. All modules, except the procedural memory module, are accessed through a specific buffer. Buffers represent a strictly limited working memory: each buffer can hold one element at maximum. Working memory is not separated from long term memory, but is the portion of declarative knowledge that is currently active, and represents the current state. Declarative knowledge is represented in ACT-R as *chunks* of information; procedural knowledge is represented by if-then rules called *productions*. For a visual overview of the ACT-R architecture, see Figure 2.2.

ACT-R's qualitative nature is represented by the production system it embeds, its quantitative nature by a set of massively parallel processes that can be summarized by a number of mathematical equations. These subsymbolic equations control many of the symbolic processes.

Models developed in ACT-R are goal-directed. Its processing consists of moving from an initial state to a specified goal state, which is encoded as a special type of declar-

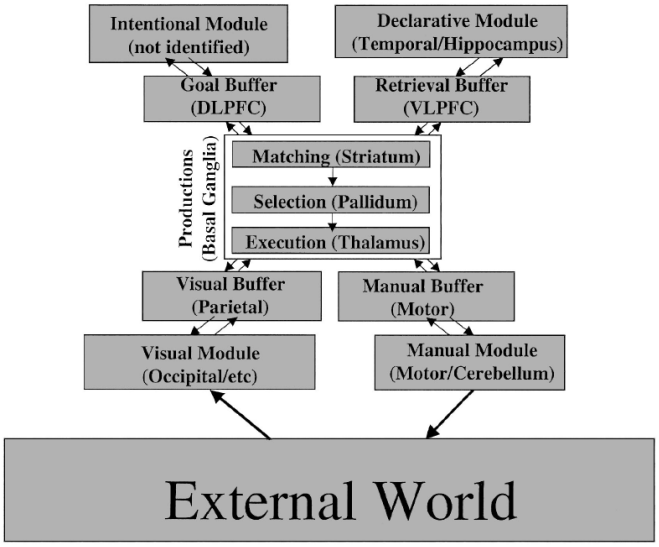


Figure 2.2: The organization of information in ACT-R 5.0 (Anderson et al., 2004))

ative memory element. This movement takes place when productions fire. A production responds to some goal, can retrieve information from declarative memory, and possibly takes some action or sets a sub-goal. Created sub-goals represent intermediate steps toward the end goal, form a hierarchy, and have to be satisfied in a bottom-up manner.

A maximum of one production can fire at a single time, so when the conditions of more rules are met, a conflict resolution mechanism determines which one fires. It does this using the rules' expected gain value, determined by a quantitative utility equation, and based on the rules' probability of success, their costs, and the current goal's value.

Initial knowledge, both declarative and procedural, is hand-coded by the human modeler. Also many values have to be hand-coded, such as the initial numerical parameters for the strength of productions, the cost and probability of success of these productions with respect to a goal, and the base-level activation of declarative knowledge structures. However, ACT-R assumes several learning mechanisms that tune these values in runtime. For example, the activation level of declarative knowledge elements is raised the more the element is used. Similarly, the expected gain value of productions is heightened the more often the rule is successful, e.g., when its sub-goal is achieved.

Applicability ACT-R models have been developed and validated against human data for a large number of tasks. Most of these efforts concern the modeling of memory tasks or small problem-solving tasks, like the Towers of Hanoi (Gunzelmann and Anderson,

2001). For these tasks it is shown repeatedly that ACT-R is capable of matching human performance data, including cognitive biases displayed by people. For example, Anderson et al. (1998) show that the ACT-R theory accounts for a wide variety of list memory paradigms, including the recency and primacy effect.

In recent years, ACT-R has also been used to model human behavior in military simulations. For example, Best et al. (2002) developed as part of the Virtual Training & Environments (VIRTE) program ACT-R models that control opposing and attacking forces in the MOUT domain. Following this exercise Best and Lebiere (2006) describe a general approach for developing ACT-R agents that have to interact in a three-dimensional space in real-time with humans and each other. Another military task was modeled by Anderson et al. (2004). They describe how the data of an ACT-R model practicing a simplified Anti-Air Warfare Coordinator task strikingly matches the learning and performance curve of humans. Juarez-Espinosa and Gonzalez (2004) discuss an ACT-R model of situational awareness (SA) for military command and control that can reproduce a commander's behavior. Although their model is very simple and supported by multiple information pre-processing agents, they do manage to show a relation between the modeled commander's memory, displaying omission errors, and SA performance.

Concerning the modeling of errors with ACT-R, Byrne and Kirlik (2005) model the taxiing behavior of pilots from commercial airlines, and show that a variety of decision-related errors can be correctly modeled. For this, their model assumes that when time is short, pilots use heuristics. More precise, in these circumstances the pilot models tend to rely on computationally cheaper, but less specific, information gained from experience with a wider class of situations, of which their current situation is an instance. This work resembles in some aspects the work of Schooler and Hertwig (2005) who also focus on the modeling of heuristics within ACT-R. Fotta et al. (2005) used the ACT-R architecture as basis to build their Human Error Modeling Architecture on, and provide a clear overview of the types of errors that the mechanisms embedded in ACT-R can produce. Also Lebiere et al. (1994) focus on error modeling in ACT-R and show that omission and commission errors can satisfactory be modeled.

Besides the modeling of human behavior for synthetic entities, ACT-R has also been used to successfully model intelligent tutors teaching mathematics and computer programming in high schools, see (Carnegie Learning Inc., 2009). These, and other intelligent tutors, are further discussed in Section 2.4.2.

Relevancy The ACT-R architecture is interesting for our goal of modeling cognitive agents for training simulations. First, ACT-R has been shown to be sufficiently powerful to represent individual human behavior in military situations. Although it must be noted

that in all the mentioned research projects the agents simply performed doctrine behavior, following a fixed set of rules. For example, the only variability found in the behavior rules of the attacking forces modeled by Best et al. (2002) was whether they, after entering a room, would clear the area to the right or left. Therefore, it remains to be seen how suited ACT-R is for the modeling of varied behavior as required for training (Section 1.1.5).

A second reason why ACT-R is interesting for our research objective, is that it has been extensively demonstrated that the ACT-R mechanisms give rise to many typical cognitive biases and errors. Most of the research mentioned shows that specific (phenotype) errors can be explained by the specific mechanisms (error genotype) embedded in ACT-R. An exception on this is the research on errors stemming from the use of heuristics, which have to be hand-modeled by the agent designer. In the latter case an additional mechanism is required that controls when these heuristics are used. But also in the former case it should be so that under specific circumstances of stress and fatigue the amount of errors increases. To model these kinds of aspects it is required that behavior moderators are incorporated within the ACT-R framework. Ritter et al. (2007b) discuss six theories of stress and show how four of these can be incorporated in ACT-R as overlays. An overlay is defined as 'an adjustment or set of adjustments to the parameters or mechanism that influence all models implemented in the architecture to reflect changes due to an altered mental state or due to long term changes such as development.' The two other theories were not possible to model as overlays, but require a more fundamental change in the architecture. One of the theories, named 'the task as stressor' has been modeled as part of this study, see Section 5.2. The modeling of fatigue within ACT-R is done by Gunzelmann et al. (2007). They incorporate fatigue as an overlay and show that its effects on performance in a sustained attentional task resembles human data.

Because ACT-R is a hybrid architecture, it remains to be seen how transparent the functioning of the models developed in ACT-R is. Related to this is the question how reusable the knowledge embedded in the developed models is.

To get hands-on experience with the two architectures described in this section, and to develop a sense for the kind of tasks they are suited for, we have developed cognitive software agents in ACT-R and Soar, in addition to cognitive agent models implemented in Swi-Prolog (Wielemaker, 2003) and LeadsTo (Bosse et al., 2007).

2.4 Applications of Cognitive Models

In this section we first discuss four application domains in which cognitive models can be applied to model human-like behavior. Next, we elaborate on the generation of feedback on task performance; also in this process cognitive models can play a role.

2.4.1 Human Behavior Models for Simulated Environments

Much of the research discussed so far focuses on the modeling of cognition or human behavior from the perspective of natural science, i.e., to understand it better. In contrast, here we describe four distinct research areas that develop software agents for the purpose of generating human-like behavior for specific tasks, so from a design science perspective. This division in four areas is adopted from Ritter et al. (2003) who describe four main application areas for human behavioral models: *education and training*, *entertainment*, *automated support*, and *systems analysis*. For each area we examine to what extent the general requirements of the behavior of the developed agents match with the requirements of the behavior of an agent for our goal, which is the ability to display human-like behavior at a variety of performance levels: from rational to biased.

Training and Simulation

Nowadays, a large part of military training takes places in simulated environments, for the reasons mentioned in Section 1.1. Tasks that are trained range from low-level tasks such as shooting at targets, to high-level tasks such as commanding a complex military operation. Setting up training for the last type requires many people, in particular if every entity in the training environment has to be controlled by a human. Luckily, this is often not required. In general soldiers do not operate alone, but as a team. Such a team is commanded by a single entity. When this entity is human, the members of such teams can be simulated without losing behavioral validity, since this is preserved by the human commander. Such simulated entities are generally called Computer Generated Forces (CGFs). CGFs that can be controlled by a human player are also referred to as Semi-Automated Forces (SAFs). For an overview of specific programs that are developed to create synthetic forces, see Table 2.2 in Pew and Mavor (1998). Well known are the packages ModSaf and its follow-up OneSaf. In addition, some military training systems, e.g., Virtual BattleSpace 2 (VBS2), also allow the programming of their virtual entities.

Because the human decides about the actions of the simulated entities, CGFs do, in general, not posses much intelligence or reasoning capability. This is fine for most tasks, e.g., walking from point A to point B can be done without much intelligence in most cases. However, their behavior might turn awkward due to their limited intelligence. A case in point is when there lays a minefield between A and B. No human soldier, even when ordered, would keep walking forward when all its team members get blown up before him. Unfortunately, CGFs are likely to do so. The limits in intelligence in CGFs becomes particular prevalent when their behavior is set by a human modeler before the training scenario starts, and is not controlled during training. Unfortunately, this is

exactly the situation of much training nowadays due to limited personnel.

Anja van der Hulst (TNO Defence, Security and Safety, personal conversation) trains military personnel for foreign peace-keeping missions by running them through a series of tactical scenarios developed in VBS2. This software package can simulate a variety of environments as well as human entities, and offers a scripting language to model behavior for these simulated human entities using basic, pre-defined behaviors. Unfortunately, it is proven time after time that the intelligence incorporated in these basic behaviors is not sufficient to create human-like behavior, and therefore to create a realistic training environment with according experiences. Van der Hulst discusses two cases in point, emerging from the basic behavior incorporated in CGFs to 'take cover when fired at'. The first is that these CGFs will, e.g., when fired at in the middle of the desert, run for miles and miles until this cover is reached. The second that 'taking cover' is also implemented as standing with 20 other CGF's in a market stall consisting of a cotton cloth, given that the latter is an 'object'.

These two examples highlight a critical aspect for the generation of human-like behavior in simulated environments; the correct representation of knowledge within the agent (first example) as well as in the simulated task environment (second example). The second example also touches upon the topic of whether all required knowledge should be formed by the agent on-line, or whether part of it can be put in the environment off-line. In the first case the agent should be able to conclude, e.g., that a cotton cloth is not appropriate to take cover, for which it has to have knowledge about the properties of cotton and the ones required to be a suitable cover. In the second case it is predetermined that a cotton cloth is not suitable for taking cover, and therefore the environment does not tag this object with the property 'cover against fire'.

The finding that CGFs are not able to display human-like behavior is not new. Sandercock (2004) identifies, using a Turing Test, several areas in which CGFs consistently show weaknesses when compared to human players. It is found that current CGFs perform weakly on: environment awareness, human variance, persistence, vengeance, anticipation, learning, and teaming.

Relevancy Current CGFs are in general not yet capable of displaying human-like behavior that is required for realistic tactical training, which includes capabilities such as situational awareness and anticipation. They are unable to display rational behavior, let alone more biased behavior. The military is interested in synthetic entities that do not display the mentioned weaknesses of current CGFs. Our proposal to get there is embedding valid human behavioral or cognitive models in the software agent controlling these entities.

Several other researchers have attempted to do the same. Section 2.3.2 described attempts to develop cognitive models that could control synthetic entities in Soar (Jones et al., 1999; Wray et al., 2002, 2004; Van Lent et al., 2004) and ACT-R (Best et al., 2002; Juarez-Espinosa and Gonzalez, 2004). In addition, Van Lent et al. (2004) developed human behavioral models in PMFServ. The development of JACK was spurred because of the necessity to model teams of BDI intelligent software agents for military training simulations (Lucas and Goss, 1999).

For a more general survey on the technologies and progress made in the construction and use of software agents controlling synthetic entities see (Stytz and Banks, 2003a,b; Banks and Stytz, 2003). In general it can be stated that there does not yet exists a satisficing solution for developing intelligent behavior for training simulations that meets our requirements.

Entertainment

In the last two decades the entertainment industry has put much time and effort in developing intelligent software agents for simulated environments: in particular for First Player Shooters (FPSs). FPSs are video games in which the player's on-screen view of the game world simulates that of a character. FPSs generally involve an avatar, one or more ranged weapons, and a varying number of enemies (Rollings and Adams, 2003). These enemies can be played by other humans, for example in on-line multi-player games, but are usually played by bots. Bots are pieces of software that control simulated characters in a game environment. Bots are programmed using techniques from so-called game Artificial Intelligence, which is a subset of common, academic Artificial Intelligence (AI). Typical examples of game AI are collision detecting techniques, and path finding and path planning algorithms such as A* (Hart et al., 1968). In general, the reasoning of game bots is scripted or limited to a few simple heuristics.

Dependent on the level of the human player, bots are easier or harder to beat. In any case it holds that certain abilities of bots, such as a perfect hit rate, should be toned down to give the human player a feeling of fairness. Laird and Duchi (2000) conclude from their attempt to model a human-like Quake-bot using Soar that *firing accuracy* and *movement speed* are significant determinators for the believability of the bot.

Although the firing accuracy level is constrained to preserve believability, it can still have different values. As such, it is possible to generate behavior at different competence levels. Unfortunately, these variations often do not occur within play, but are set beforehand. When bots do alter their competence level within play, this is not due to human-like aspects as stress or exhaustion. They alter their level to match the level of the human player, to increase the fun of the game (Scott, 2002).

Relevancy Many techniques used in game AI might be useful for our purpose. However, reuse is not without danger, since bots are developed for simulations that have as their primary function entertainment, and not training. For game AI it holds (Tozour, 2002):

> "Our field requires us to design agents that produce appropriate behaviors in a given context, but the adaptability of human-like 'intelligence' is not always necessary to produce the appropriate behaviors, nor is it always desirable".

For example, hacks and cheats are generally accepted as long as they increase game play, and are not obvious (Scott, 2002). An example of a common cheat is that bots know where the player is, although they have not observed that. Such cheats can easily cause unrealistic behavior, which is fine for games as long as it enhances the fun of playing. However, such behaviors might not always be appropriate for training simulations. For example, when a trainee has successfully crept up on an enemy, it is undesired that this behavior goes unrewarded because the enemy knows where the trainee is anyway.

For training simulations the main goal of modeling human-like behavior is not to increase the *fun* of training, but to increase the *realism*. It is doubtful whether knowledge concerning the modeling of human-like behavior in games can aid this increase in realism. For example, game AI has not been been used to model biased behavior. For a survey of commercial game technologies related to behavior modeling and their relevance for military simulations, see Diller et al. (2004).

Automated Support

The increase in system technologies over the last decades has resulted in an increase of information that is available at any time to an operator of such a system. Unfortunately, more is not always better. Too much information can lead to a state of *information overload*. On top of this, information might be incomplete or uncertain, which further complicates the decision-making process. As such, there has been an increasing demand for automatic support of system operators.

Systems capable of generating support are usually referred to as intelligent decision-support systems. The support they provide varies: some take over the task, others prepare materials or information, yet others modify the display to help distinguish between alternatives, or to make performing actions easier. Some of this support can be delivered by a software agent in the form of an intelligent assistant. An example of such an intelligent assistant developed using the COGNET architecture is described by Weiland et al. (1998). Their Naval Surface Fire Assistant helps a human gun commander to utilize a gun system, and is understandable to its human supervisor.

For generating support, it is minimally required that the intelligent decision-support system incorporates an *expert model* of the task to ensure that its recommendations are justified and useful. However, by itself, this is not sufficient. Current decision-support systems are often not used efficiently, because the operators lack confidence in the recommendations of the system (Moulin et al., 2002). To increase confidence, the system should be able to convince its user that its suggestion is justified and useful. For this it is important that the system incorporates a *user model*. A user model can be defined as 'a model that incorporates information about a system user to provide services that supports its demands' (McTear, 1993). Neerincx (2007) states that intelligent decision-support systems for complex, military tasks have to acquire and maintain knowledge of the cognitive and affective load of the tasks and situation, and the capacities of the user to cope with this load. By means of cognitive and affective load models adequate decision-support can be given that supports shared situation awareness, trust and scrutability.

Relevancy Expert and user modeling is relevant for training systems: feedback that is provided to the user of a training system (a student) should be based on an expert model, and tailored to the specific user. More specific, for adapting instructions and feedback to the individual student, a *student-model* that incorporates the knowledge of the student is required.

Therefore, knowledge developed within the field of intelligent decision-support systems concerning the modeling of expert and user models is relevant to our research objective. However, this knowledge might not be directly applicable and reusable. Supporting decisions in minimal time might require a different type of expert and user model than task training. In addition, the development of users models for decision-support is a younger research area than the development of user models for training, which makes it more likely that the former can learn from the latter. In Section 2.4.2 we elaborate on methods with which expert and student models can be formed that are developed within the research field of intelligent tutoring systems.

Systems Analysis

A last field that realistically models human task performance in simulated environments is engineering. The field of engineering has created multiple models of users of their systems, often on a complex operator level. These simulated operators are used to evaluate proposed human-machine designs and operator procedures.

Several general engineering-based architectures for the modeling of human operators have been developed, see Pew and Mavor (1998) for an overview. In general the main focus of these engineering-based architectures is the modeling of the human-machine

interaction part. Therefore, they are more concerned with modeling the psychomotor tasks of humans than with modeling cognitive processes. The main reason why it is possible for the model developers to abstract from internal cognitive mechanisms is that the tasks for which they model operators are commonly well-defined, procedural tasks.

Nowadays, this field starts to realize the potential of using cognitive models to evaluate systems for a wider range of tasks than currently looked at, i.e., ill-defined, complex tasks instead of well-defined, procedural tasks. For a paper promoting the synergy of research into cognitive modeling and human-machine interaction, especially concerning the evaluation of user interfaces, see Ritter et al. (2001). Two examples of such research efforts are described by Amant et al. (2007) and Ryder et al. (1998). Amant et al. describe a cognitive model developed in ACT-R for the evaluation of cell phone menu interfaces, while Ryder et al. describe a cognitive model developed in COGNET for the evaluation of an integrated telephone services workstation interface.

Relevancy Human-like operators developed for system analysis could in principle be used to play the operator role in a training simulation. Unfortunately, these models are often task specific and connected to a simulation of the domain they are developed for. This makes it hard to gather which aspects are relevant and reusable for our domain.

In addition, the developers of operator models are in general more interested in a model that is usable and approximately correct, than in modeling the detailed internal mechanisms giving rise to behavior (Ritter et al., 2003). A case in point is that specific operator models, and engineering-based architectures in general, do not model any type of errors.

Moreover, current operator models are often developed for tasks that are not at the level of complexity sought after. The knowledge developed in this field concerning the modeling of expert behavior for well-defined, procedural tasks is likely to be reusable for training of such tasks. But for modeling operators in complex, open environments that perform tactical tasks and make realistic mistakes, the field of systems analysis does not offer a solution.

2.4.2 Feedback Generation for Simulated Environments

Previously, we pointed out that an important factor in task training is the generation of feedback on the student's task performance (Bosch and Riemersma, 2004). Training opportunities would increase significantly when this feedback could be generated by a system instead of a human. Automatic generation of feedback is one of the focuses of the research field of intelligent tutoring systems (ITSs), see e.g. Polson and Richardson

(1988). Other focuses are the automatic construction of an individual-tailored curriculum, and the ability to answer questions about the exercises or domain in general. For a clear overview of all the capabilities an ITS can possess, see VanLehn (2006).

A system capable of generating feedback should foremost be able to diagnose the task performance of a student. This entails assessing the actions of the students, and deciding whether or not they are appropriate for the current task. Moreover, the system should be able to provide the student with a motivation for its diagnosis.

In this section we aim to shed light on the current state-of-the-art concerning feedback-generating systems, and on the relevancy of that work for generating feedback on open complex tasks in military simulations. After distinguishing various types of feedback, we discuss which feedback type is required, or suitable for, which task type. Subsequently, we elaborate on the models and techniques that are currently used by ITSs to diagnose a student's task performance and to generate appropriate feedback.

Feedback and Task Types

Three types of feedback can be distinguished that are relevant in the context of simulation-based training systems (Mioch et al., 2007). They differ in the types of information that they take into account, and in the level of sophistication of the feedback they generate.

- **Result-based feedback**: This type of feedback is based solely on the result of the task behavior of the student. Feedback is generated by comparing this result with the correct result, which is often hard-coded and formulated beforehand by domain experts. The feedback states only whether the student has completed the task successfully, and if not, which result would have been correct.

- **Model-based feedback**: This type of feedback is not only based on the result of the student's behavior, but also on contextual knowledge of the simulation environment and explicit task knowledge. Feedback is generated by reasoning about the result of the student and why it was good or false, for which it uses an expert model and the task circumstances.

- **Cognition-based feedback**: This type of feedback is also based on the student's result, the context of the task, and explicit task knowledge, but additionally takes a student model into account. This student model tracks behavior of the student over time and makes it possible to infer the cognitive strategies of the student. Using this extra knowledge, feedback can be generated not only on the final result of the student's behavior, e.g., the selected action, but also on the process, e.g., how he or she selected this action.

These types of feedback can be further illustrated by an example from a traditional domain of feedback-offering training systems: mathematics. In mathematics, tasks have to be trained for which the interpretation of the result is deterministic: a result is either correct or false. An example of such a task is addition. The correct answer for adding 9 to 32 would be 41. In case a student arrives at the result of 31, the *result-based feedback* would entail something like: 'No, your answer is wrong, the correct answer is 41.' The *model-based feedback* would for example be: 'No, your answer is wrong. The correct answer is 41. You calculate this by first adding 2 to 9 which gives you 11, after which you note down 1 and subsequently add the left over 1 to 3, which gives you 4.' On the other hand the *cognition-based feedback* might entail: 'No, your answer is wrong. The correct answer is 41. After adding the 2 to the 9 and noting down 1, you probably forgot to carry over the remaining 1 and add it to the 3.'

Another division of feedback types is given by VanLehn (2006), who distinguished two types: *minimal* and *error-specific* feedback. Minimal feedback is another word for result-based feedback, while error-specific feedback refers to both model-based and cognition-based feedback.

Most ITSs have been developed for the training of tasks in well-structured and small domains, like the addition example above. Such domains have little indeterminacy and are relatively closed. Therefore, they can be represented by a small number of rules. Moreover, the tasks that are trained are often individual, procedural and static. For these types of tasks and domains it is relatively easy to deduce all three types of feedback, although in general the feedback is limited to result- or model-based feedback. Feedback can be based on expert knowledge in the form of rules or constraints that can unambiguously, due to the task's nature, determine the correctness of certain actions in certain states.

More challenging is the generation of feedback on student behavior in simulated environments. The types of tasks that are commonly trained in simulation-based training systems share few of the properties mentioned above. In general, these tasks are dynamic, open, and complex, since they take place in the (simulated) world which shares these properties. This entails that the task domain cannot be represented by a small number of rules. A task is called *dynamic* when it takes place in an environment that changes constantly and independent of the task performer. It is called *open* when multiple correct options exist to reach its goal. It is called *complex* when in every situation multiple actions are possible, that lead to new situations in which again several actions are possible. The dynamic nature of a task makes that for evaluating a student's performance not only its actions, but also the timing of its actions needs to be taken into account. The open and complex nature of a task makes that no single correct behavior exists at any moment,

which makes performance diagnosis hard. Moreover, the correctness of a certain action often depends on the correctness of the actions that follow it. In other words, on the strategy that is followed by a student.

ITSs that have been developed for training dynamic, open, and complex tasks generally limit themselves to providing model-based feedback (Stottler and Vinkavich, 2000). However, it would be beneficial if feedback is provided at the level of cognition, e.g., on the appropriateness of the followed strategy. For this, cognitive models can be used.

Ryder et al. (2000) compare several uses of cognitive models in ITSs, and focus on expert performance models and instructional agents in particular. The models they review are all based on the cognitive modeling framework COGNET and provide feedback at various levels. For example, the described Advanced Embedded Training System (AETS) (Zachary et al., 1998) provides model-based feedback, while the feedback provided by the instructional agent embedded in their Electronically Assisted Ground Based Learning Environment (EAGLE) is sometimes cognition-based.

Other well-known research on cognitive models for training cognitive tasks is performed by Anderson and colleagues (Anderson et al., 1990, 1995). They successfully implemented several *cognitive tutors* that were developed around task-specific cognitive models in the form of production system models in ACT. The cognitive task models and thus tutors were developed for were well-defined tasks, namely LISP programming, high-school geometry, algebraic manipulation, and word problems.

Required Models

The generation of feedback to a student on his or her task performance has much in common with the generation of support to a user on the basis of his or her performance. For this, it is first of all important to incorporate an accurate and complete task knowledge model, i.e., a so-called expert model. Moreover, it is important to tailor the feedback to the user, for which a student model is required. Last, an instructional, also called pedagogical, model should be incorporated to determine the form of the feedback (Polson and Richardson, 1988).

Expert Model Anderson (1988) identifies three approaches to encoding expert knowledge. The first is independent of the way it is present in human intelligence. For example, in some domains it is possible to build a mathematical model whose subsymbolic equations deliver the same results as reached by humans after symbolic reasoning. The second is by building a standard (rule-based) expert system. The knowledge is extracted from a human expert, but the representation and the way it is applied does not necessarily have to correspond to the way it is represented and applied by humans. The third

possibility is to build a cognitive model that simulates the way humans use and apply knowledge. On the whole, it is important that the feedback generating system incorporates a knowledge representation that facilitates the abstraction from low-level knowledge to high-level knowledge, since the latter is often required for feedback (Gomboc et al., 2005).

Which specific approach to encoding expert knowledge should be followed depends on the type of feedback that has to be based on it, and the method that is used to generate this feedback. For generating result-based feedback the 'final' observable result of a student's task behavior has to be compared with that of the expert. How this expert result is determined is unimportant, so the first method to encode expert knowledge suffices. For generating model-based feedback, task knowledge and contextual circumstances have to be taken into account in addition to the final result. This can be done when expert knowledge is encoded following the second approach. The rules of the expert model embed the task knowledge, and give insight in the relationship between the current context and the correctness of the result of the student's behavior. For generating cognition-based feedback, the cognitive strategies followed by the student have to be addressed as well. In order to do so, expert knowledge should be encoded as a cognitive model, because such a model provides the required insights in the cognitive reasoning processes for generating this type of feedback.

Student Model Feedback should not be based solely on the expert model, but should be adapted to the individual knowledge state of the student. Therefore, the feedback generating system needs to keep a model of the student's knowledge, ideas, and beliefs. This model should be dynamic since these aspects change constantly during task execution.

VanLehn (1988) introduces two types of student models. The first are student models that can only represent which task conceptions the student is missing. Such models are also called overlay models, since they represent the overlay of the student's knowledge with the domain knowledge, i.e., it models the student's knowledge as a subset of the domain knowledge. A system taking this approach requires a knowledge base that contains the domain knowledge, i.e., the facts, rules, and competences that the student has to learn, and the ability to mark the content that the student has mastered. The second type of student models are those that can represent besides missing conceptions also the student's misconceptions. The way in which these can be diagnosed is the topic of the next section, which describes several diagnostic techniques.

An important factor for diagnosing a student's current knowledge-state is the level of input that is available to the diagnostic module, also called the *bandwidth* of the information (VanLehn, 1988). Broadly stated, three categories of information bandwidth

exist. The lowest category is **final states**. This category is applicable when the diagnostic module only has access to the final state of the task execution process, i.e., its result. The second category is **intermediate states**, which is applicable when besides the result of the task execution process also intermediate steps toward reaching this result are available. The third category is (approximate) **mental states**. For this category to apply, the diagnostic module should know about the mental states the student traverses while performing the task. These states are obviously not directly observable, so should be inferred. The derivation of a student's mental state from his or her behavior is called *cognitive diagnosis*.

Techniques for Diagnosing Student Performance

For a detailed overview of nine diagnostic techniques, each of them useful for a specific type of information bandwidth system, see VanLehn (1988). Here we elaborate on the two techniques that have surfaced in the last decades for generating cognition-based feedback, as this is the type of feedback that is required for open, complex military tasks. These two techniques are **model-tracing** and **constraint-based modeling**.

Model-Tracing Model-tracing is a technique that follows a *process-centric* approach. The diagnostic module tries to infer the (erroneous) process by which a student arrived at a solution, and bases its feedback on this. Famous examples of ITSs that took a model-tracing approach are the cognitive tutors developed at the Pittsburgh Advanced Cognitive Tutor Center at Carnegie Mellon University (see, e.g., Anderson et al., 1990, 1995). This group also developed Cognitive Tutor Authoring Tools (CTAT) that assist in the creation and delivery of ITSs based on model tracing (Aleven et al., 2005). Zachary et al. (1999) also followed this approach for their Advanced Embedded Training System (AETS).

An ITS that embeds the model-tracing approach incorporates an executable cognitive expert model that possesses correct declarative and procedural domain knowledge. Using this knowledge, the system can simulate adequate task behavior. In addition, the system incorporates false domain knowledge, which can simulate typical mistakes made by students. Moreover, the ITS has the capability to trace the student's behavior step-by-step, and to compare these steps with the correct and false steps that are dynamically generated by the expert rules and so-called buggy rules, respectively. Feedback on the student's process can be generated at every step, based on the rule with whose result the student's behavior matches.

The main advantage of model-tracing is that following this technique, error-specific feedback can be generated. Another advantage is its modularity: buggy rules are easily added. However, relying on buggy rules to generate feedback also has several disadvan-

tages. First, feedback can only be given when the student's behavior matches that of one of the modeled rules. Hence, it is important that all possible errors are represented by buggy rules, which can be a hard and time-consuming effort. In addition, it is possible that the student's behavior matches that of multiple rules. In that case, it is a complex task to ascertain what the student's knowledge state is.

Research, among others the projects mentioned above, has shown that the model-tracing methodology of following a student's behavior step-by-step, possibly generating feedback at every step, is applicable for well-structured, procedural tasks. However, whether this is also the case for open, complex tasks is yet unclear.

Constraint-Based Modeling Constraint-based modeling is a technique that follows a *result-centric* approach. The diagnostic module bases its feedback solely on the final-state the student arrived at, independent of the process that led him or her there. Nonetheless, this feedback is different from result-based feedback: the diagnosis does not only say whether this final-state is correct or what the final-state should have been, it also gives feedback on the type of error the student has made.

Constraint-based modeling is based on the idea that students learn from performance errors, and that those errors are the result of declarative knowledge that has not yet been internalized by the student. This approach assumes that correct solutions all satisfy the general principles of the domain, which can be described by *constraints*. When the student's final state violates one these constraints, this signals an error. The violated constraint gives direction to the incomplete or incorrect knowledge the student has, which can be used as basis for feedback.

Well-known examples of constraint-based tutors are those developed at the Intelligent Computer Tutoring Group at the University of Canterbury (see, e.g. Mitrovic et al., 2007). This group also developed a tool that assists with the creation and delivery of constraint-based tutoring systems, named ASPIRE (Mitrovic et al., 2006). The constraint-based modeling approach is also used by Ryder et al. (2000) to construct the instructor agent embedded in their Electronically Assisted Ground Based Learning Environment (EAGLE) for the training of UAV operators, see page 57.

Comparison of Techniques A significant advantage of constraint-based modeling over model-tracing is the smaller required development effort. There is no need to specify an executable expert model, nor a complete set of buggy rules. Moreover, the student is not constrained in the process leading to its results. As long as its final-state does not violate a constraint, the student is free to think of new solutions. However, the fact that a student's process is not constrained has raised discussion on the applicability of

constraint-based modeling to the training of procedural tasks (Kodaganallur et al., 2005). Moreover, Kodaganallur et al. question its capability to generate error-specific feedback, in specific whether it is possible for all types of tasks to specify abstract constraints that must always hold for a solution. This last point is relevant to our research objective. For training open, complex tasks in which not the outcome of a decision is of main importance, but the reasoning-process leading to this decision, it might be difficult to specify such constraints. For extensive comparisons of the model-tracing and constraint-based modeling techniques and discussions on their suitability for various types of tasks and domains, see Mitrovic et al. (2003); Kodaganallur et al. (2005); Mitrovic and Ohlsson (2006); Kodaganallur et al. (2006).

2.5 Conclusion and Prospect

In this chapter we introduced and discussed research from various research fields. We introduced this wide variety of research topics because the research described in the rest of this dissertation is related to it all. We took the freedom given by the general topic of 'cognitive models for training simulations' to perform a PhD study spanning several research topics and research fields.

In the coming chapters we address five different research topics. Each chapter embeds papers that are all but one published, and starts with an introductory section. In these introductions we link the papers within the chapter to each other and place them within the general research framework as introduced in this chapter. When required, we also elaborate on more specific related work than described in the current chapter.

Although the topics addressed are quite diverse, the same research thread is prevalent in them all: using *natural science knowledge* concerning cognition and education and *design science knowledge* concerning artificial intelligence and software engineering to construct models that, when implemented in software, form agents that can display, or provide feedback to, human (biased) behavior.

Chapter 3

Belief Component

3.1 Introduction

Previously, we introduced the *regulative research cycle* that describes the general course of action followed in this study (Section 1.3.2). The cycle consists of five phases. First, a *research problem* is defined in the form of a discrepancy between the aspects of human behavior that existing methods can model and the required aspects. Next, it is *diagnosed* what this discrepancy exactly entails, and which (cognitive) properties the to-be-developed method should be able to model. Then, a *solution* to the problem is proposed, for which we use theories on the underlying mechanisms of the required behavior.

These first three phases of the regulative research cycle are shortly discussed in the introduction and related work sections of the four research papers embedded in this chapter. However, due to the limited space available there to discuss these three phases, we elaborate in this introduction section on which behavior is required for our research objective, what the current state of the art is for modeling such behavior, and how we propose to model it. The last two phases of the research cycle, the *implementation* of the solution proposed and the *evaluation* of this implementation, are described in the papers.

In the beginning of this dissertation we elaborated on the decision to model the cognitive processes underlying situational awareness (SA, see Section 1.4). SA is a state of knowledge; situational assessment is the process that achieves, acquires, or maintains that state (Endsley, 1995). The maintenance of a knowledge state is considered to be a cognitive capability, see Section 2.2.1. Gordon (2005) labels this capability 'Knowledge and Inference' and defines it as 'models of how people maintain and update their beliefs in the face of new information'. Langley et al. (2006) label it 'Reasoning and Belief Maintenance' and define it as 'the ability that lets an agent augment its knowledge state'.

The ability to maintain beliefs is a fundamental capability underlying human cognition, and should therefore be modeled. Previously, we explained our decision to model cognition using symbolic mentalistic notions such as beliefs (Section 2.3.1). Beliefs are used to denote an agent's knowledge state, the definitions given above by Gordon (2005) and Langley et al. (2006) also mention both beliefs and knowledge.

In this chapter we develop, implement, and validate a belief framework for cognitive agents that will enable them to maintain their beliefs. Because we want cognitive agents to be able to display varied behavior, the framework supports rational as well as biased belief maintenance. Military situational assessment tasks are used as research context.

3.1.1 Existing Methods for Belief Maintenance

In this section we describe existing methods for maintaining beliefs. First, methods stemming from Artificial Intelligence are discussed, next methods developed in Cognitive Science, and last the methods incorporated in Soar and ACT-R. In the next section we discuss the relevancy of the introduced methods to the current research context.

Belief Maintenance within Artificial Intelligence

To ensure that an intelligent agent selects the proper intentions (plans) for fulfilling its desires (goals), its set of beliefs concerning its own status and that of the world should be consistent and correct. Many researchers in AI have investigated how to maintain a correct and consistent belief database when an agent receives new information.

Belief maintenance actually combines two capabilities: the handling of information received from different sources, also referred to as *source-integration*, and the handling of information received over time. For an overview of research into integrating pieces of information issued from several sources, see Bloch et al. (2001). For handling information received over time there exist, broadly stated, two generic approaches: *belief revision* vs. *belief updating*. Belief-update mechanisms assume that upon receiving new information relating to a changed world, the agent's belief set has become obsolete. They therefore alter the agent's beliefs in such a way that its belief base incorporates the change. Belief-revision mechanisms assume that new information as well as the agent's current belief set are about the present, although the latter might be less reliable. When new information is inconsistent with the information already stored in the agent's belief base, they restore consistency by changing the old information set as minimally as possible.

How the beliefs of an agent can be changed depends on the agent's belief type. The simplest type is the binary one: in that case an agent believes the stored information to be either true or false. More complex are the belief types that represent the fact that an

agent holds a statement to be more or less true. Such belief types can be implemented by attaching a degree of belief to the information stored. This degree can be a numerical value or a conceptual label, e.g., 'perhaps' and 'likely'.

The formation and interpretation of the agent's degree of belief in stored information can vary in a number of ways. For example, the degree of belief attached to information elements can be absolute, or it can depend on the other information elements the agent holds, in which case it is relative. In addition, the degree of belief can denote the (un)certainty about the total belief, e.g., I am quite sure (with p = 0.9) that the speed of contact1 is 20 knots, or it can denote the dispersion of the value believed: I believe that the speed of contact1 is about 20 knots (20$\pm$2 knots). Moreover, the attached degree of belief can represent the probability of that information being true, or it can represent the plausibility of that information (Dubois and Prade, 2001). In the former case a single value denotes how probable information is, i.e., how sure the agent is of the truth value of the belief. In the latter case two concepts are used: the possibility and the necessity of information, which together denote a range of belief.

Belief maintenance techniques have been developed for all types of belief. Belief maintenance techniques that operate on binary beliefs are activated when an agent receives information that is inconsistent with its current beliefs. Frequently, such an inconsistency is resolved by deleting some of the beliefs that cause the inconsistency. This strategy is for example implemented in the dominant AI theory on belief maintenance for binary beliefs, the so-called AGM postulates. These postulates formulate properties that an agent should satisfy in order to be considered rational (Alchourròn et al., 1985).

The AGM postulates illustrate when AI considers behavior to be intelligent: when it is rational. In contrast, we are interested in modeling human-like behavior including human-like belief maintenance, which is not always rational. In addition, the deletion of beliefs, which entails that an agent forgets what it believed to be true, is not human-like, nor desirable (Byrne and McEleney, 2000). Also the fact that many belief revision techniques offer a single way to resolve inconsistencies is debatable, since it is shown that the nature of the beliefs forming an inconsistency influences the way humans deal with them, see the following section.

Also for graded beliefs a wide variety of belief maintenance techniques have emerged. Within AI techniques are based on, e.g., probability theory, fuzzy logic, possibility theory, and Bayes's theorem (Bayes, 1763). Belief maintenance techniques based on Bayes's theorem are also frequently used by Cognitive Science, because they are suited for modeling normative, rational belief maintenance.

For example, *Bayesian inference* is a technique based on Bayes' theorem that can be used to implement belief maintenance for belief types of which the attached degree

denotes the probability of that information being true. Bayesian inference infers the probability that a hypothesis (information) may be true based on (observed) evidence. The Bayesian inference process takes as input a hypothesis with a certain numerical degree of belief and some new information, and outputs a new degree of belief that takes this evidence into account. In intelligent agents this mechanism can be used to control the collection of evidence; information of which it is known that it is strongly consistent or inconsistent with a hypothesis might be especially important to check. Starting from the hypothesis that humans also have the intention to collect information to check their beliefs, this control process can be subject to biases (e.g., the confirmation bias, see Section 2.2.2). Although easy to understand, Bayes theorem has one major disadvantage: for its application the probability of the hypothesis before the new evidence, and the probability of the occurrence of that evidence given the hypothesis, need to be specified. In the case of open, dynamic, and complex tasks this entails the specification of a multitude of probabilities, with as main problem that many of them are likely to be unknown.

Another technique that can be used to implement belief maintenance by calculating the probability of an event (information) by combining pieces of evidence is the Dempster-Shafer theory, developed by Dempster (1968) and Shafer (1976). The Dempster-Shafer theory (D-S theory) is a generalization of Bayes' theorem, with as main computational advantage that not all prior and conditional probabilities need to be specified: missing information is simply not used. Unlike Bayesian inference that is used for belief types of which the degree denotes the probability that that information is true, D-S theory is applicable to belief types of which the degree denotes the possibility of that information being true. For this, the degree of a proposition is represented as an interval $[Bel, Pl]$. Bel is interpreted as the degree of belief and Pl as the degree of plausibility.

D-S theory is often used to integrate information from different sources. The so-called *Dempster's rule of combination* is used to integrate beliefs corresponding to independent pieces of information. This rule emphasizes the agreement between multiple sources, and ignores all the conflicting evidence through a normalization factor. A disadvantage of using the D-S theory is that it requires all evidence to be independent. Although in specific cases this might be realistic, it is not realistic for a cognitive agent's beliefs about information from different sources which might be all biased by, e.g., an expectation about the situation. So, D-S theory might be applicable to model perfect, rational source integration, but is less applicable for modeling human source integration.

Belief Maintenance within Cognitive Science

AI pursues a belief maintenance model that enables cognitive agents to maintain their beliefs in an 'optimal' way, i.e., rational and computationally feasible. Cognitive Science

on the other hand pursues a model that validly represents the way in which humans maintain their beliefs, which might not be rational. In this section we first shortly discuss some results of experimental research on human belief maintenance. Next, we discuss several existing models on human belief maintenance developed by cognitive science.

Human belief maintenance is typically studied by presenting subjects a categorical statement (e.g., A is true) and a conditional statement (e.g., if A then B). Next these subjects are confronted with new information (e.g., not B), which is inconsistent with the information that can be derived from the categorical and conditional statement (in this case B). It is then checked whether and how the subjects revise their beliefs in order to maintain a consistent belief set.

Elio and Pelletier (1997) found that humans tend to revise the conditional statement over the categorical statement. Dieussaert et al. (2000) refined Elio and Pelletier's belief revision experiments, and conclude that the more certain humans are about the conditional, the less they are prepared to reject it, so the initial *strength* of the belief in the conditional is important for its revision. Also the *source* of beliefs is important for their revision. Mercier and der Henst (2005) show that humans are less likely to revise a set of beliefs obtained on their own when it is contradicted by communicated information, than to revise a set of beliefs obtained by communication when it is contradicted by information obtained on their own. In addition, they show that the potential manipulativeness of the communication source influences the degree to which existing beliefs are endorsed and the likeliness that beliefs are revised. Byrne and Walsh (2002) claim that the way humans revise beliefs depend on whether they made a modus ponens inference (from A to B) or a modus tollens inference (from not B to not A). In the first case they tend to revise their belief in the categorical statement, in the latter case they revise their belief in the conditional statement.

Many of the studies into human belief revision conclude that often humans do not end up disbelieving one of the statements, but rather revise them, e.g., extend the conditional rule with an extra condition or an exception (Walsh and Sloman, 2004). Hasson and Johnson-Laird (2003) stress in their paper that humans are able to reason about their beliefs and that they have multiple strategies to recognize the inconsistencies and that these can yield different belief revision patterns. Despite this, some general tendencies can be distinguished, e.g., the previously introduced order effects (Section 2.2.2). The fact that the order in which information is received influences the final belief of a person is a typical example of *judgment* bias (Section 2.2.2).

Cognitive Science has developed various models of human belief revision in order to get insight in the cognitive processes underlying the experimental results found. Hogarth and Einhorn (1992) present the belief-adjustment model that displays order effects in

belief maintenance, and accounts for them as arising from the interaction of information-processing strategies and task characteristics. In their model, all the information an agent has is represented by its degree of belief in the pieces of information, and can be written as $S_k = S_{k-1} + w_k[s(x_k) - R]$ with:

S_k = degree of belief in some hypothesis, impression or attitude after evaluating k pieces of information ($0 \leq S_k \leq 1$).

S_{k-1} = anchor or prior opinion. The initial strength of belief is denoted S0.

$s(x_k)$ = subjective evaluation of the kth piece of information.

R = the reference point or background against which the impact of the kth piece of information is evaluated.

w_k = the adjustment weight for the kth piece of information ($0 \leq w_k \leq 1$).

The model facilitates for different outcomes of the subjective evaluation of the kth piece of information depending on whether information is processed in a Step-by-Step or End-of-Sequence manner, which are two different information-processing strategies. The belief-adjustment model has been shown to be able to replicate, as well as predict, belief maintenance effects based on the order of beliefs and task characteristics. This has been shown among others in tactical decision-making (Adelman and Bresnick, 1992; Adelman et al., 1993; Zhang et al., 1998), although contradictionary results have been found as well (Adelman et al., 1996). Hogarth and Einhorn's belief-adjustment model accounts for order effects by incorporating graded beliefs and an *anchoring and adjustment* process, which is one theory of belief maintenance.

Thagard (2000) holds a different theory of belief maintenance: Thagard claims that much of human cognition, including human beliefs and inferences, can be understood in terms of *coherence as constraint satisfaction*. His theory of coherence operates over a set of representational elements (e.g., propositions representing binary beliefs) which can either fit together (cohere) or resist fitting together (incohere). If two elements p and q cohere they are connected by a positive constraint $(p, q) \in C^+$, and if two elements p and q incohere they are connected by a negative constraint $(p, q) \in C^-$. Furthermore, constraints are weighted, i.e., for each constraint $(p, q) \in C^+ \cup C^-$ there is a positive weight $w(p, q)$. Coherence maximization involves the division of the set with all elements into an accepted (A) and a rejected (R) set in such a way that a maximum weight of constraints is satisfied. Thus, the higher the summed weights of the satisfied constraints, the more coherent the solution to the coherence problem is. A question remains how such an optimal coherent set can be found. One of the options Thagard proposes is a connectionist algorithm that uses an artificial neural network to assess coherence.

Thagard's theory of explanatory coherence in implemented in the Echo model (Thagard, 1992), which is a model of abductive reasoning and incorporates binary beliefs. Although Echo captures many aspects of human abductive reasoning, it does not account for order effects. Wang and Johnson (1998) introduce UEcho as extension of the Echo model with the motivation that Echo fails to sufficiently manage the uncertainty in abduction. UEcho incorporates a dynamic processing mechanism for belief revision and graded beliefs, and does display order effects. Later, Wang et al. (2000) further extend UEcho to embed a distinction between the probability and confidence of information. This extension was required to model human data in an military situational assessment task. Wang et al. used a modified version of the CIC (Combat Information Center) as task domain. In the CIC task the goal of the participant, acting as a commanding officer of a naval ship, was to collect two pieces of information sequentially about an aircraft in the radar area and accurately identify its intention. Wang et al. (2000) show that both the experimental results and the UEcho modeling results display order effects in belief revision at the early stage of training when the confidence level is low, but that these effects tend to disappear later when the confidence increases.

Paglieri (2004) offers a quite different approach to belief maintenance: he proposes a model based on a distinction between *data* (information stored by an agent) and *beliefs* (information accepted as reliable). Paglieri actively opposes the common AI approach of maintaining binary beliefs by throwing away inconsistent ones and states:

> "Belief change should not be understood as a mere process of gathering new information and removing old ones from a given knowledge base, but rather as a *two-layered dynamic*, which involves both changes in the available data (due to information update, inferential reasoning and oblivion), and the resulting outcomes in the selection of acceptable information (i.e. beliefs) - plus changes in the selection process itself, since different agents might apply different selection strategies (e.g. skeptical vs. trustful agents), and the same agent might modify its strategy according to context (e.g. preferring caution in unknown situations, while behaving more confidently in familiar domains.)"

The distinction between data and beliefs enables the agent to preserve all the relevant information, including contradictionary ones, but at the same time to select a subset of them as beliefs where it can base its decisions and actions on. Belief change will typically involve an update in the data and a modified selection of this data as beliefs. So it might be that a belief is discharged, but this does not cause the forgetting or deletion of the corresponding datum. However, Paglieri's model does not account for the storing of what was previously believed, so when a belief is discharged, it is forgotten that that datum was once believed.

The last model of belief maintenance presented in this section explicitly takes into account the sources of the beliefs while maintaining them. Castelfranchi (1997) argues that the sources of beliefs are remembered and play an important role in their acceptance and revision. For this, Castelfranchi distinguishes three types of beliefs: a belief about the source (S-belief) that together with a belief about the reliability of the source (R-belief) leads to a resulting belief (O-belief). Perception, communication, and inferential processes are all possible options for a belief's source. We adopted a similar approach, incorporating the same types of belief sources and that their reliability influences the acceptance of the belief, in the belief framework introduced in Section 3.2.

Belief Maintenance within Integrated Architectures

The cognitive architectures developed in Cognitive Science often embed a single, or no belief maintenance mechanism at all. Frequently, the modeler is left with the task to maintain a consistent belief base. For example, Soar (see Section 2.3.2) supports two types of working-memory elements (WMEs): WMEs created by an *operator* and WMEs created by an *elaboration* rule. A WME that is created by an operator will stay in working memory (WM) until an explicit change is made. A WME created by an elaboration only exists as long as the conditional part of the elaboration matches the content of WM. This aspect of Soar is referred to as its Truth Maintenance System, because it automatically updates beliefs about the world. However, there is no restriction on the WMEs that can be posted in WM by elaborations and operators given the currently present WMEs. It is therefore possible to believe, e.g, that a surface radar contact is hostile as well as friendly: it is up to the modeler that such a situation does not occur.

ACT-R (see Section 2.3.2) embeds a specific mechanism for retrieving beliefs in WM. All beliefs stored in long-term memory (LTM) have an activation-value that reflects their use in the past (frequency and recency) and their current 'relevancy' in the sense how related they are with other currently active chunks (e.g., in the vision buffer). When a query is posed to LTM, only one belief can be retrieved which is the one that matches the request and has the highest activation. So, although it is possible that inconsistent beliefs exist in LTM, only one, thus consistent, belief is retrieved. Which belief is retrieved is determined by the belief's activations, so it is not possible to actively revise beliefs, let alone in various ways.

Both Soar and ACT-R do not lay down that the knowledge of an agent receives a degree of belief, nor that its specific source is represented. However, because the slots in the chunks of ACT-R and in the WMEs of Soar are undetermined, it is possible for the modeler to fill these slots with values denoting these aspects.

3.1.2 Selecting an Approach

All the models and techniques introduced in the previous section have one thing in common: they are designed to bring about a specific type of behavior. For example, the AGM postulates (Alchourròn et al., 1985) ensure 'rational' belief revision, while the belief-adjustment model of Hogarth and Einhorn (1992) ensures the emergence of order effects in belief revision. Which model or technique should be used to model belief maintenance depends on what is required of its behavior and its scope. Many models and techniques have a small scope and are only fitted to bring about a specific type of behavior suited for that narrow domain.

A main requirement for using cognitive agents in training simulations is that they are capable of producing varied behavior. So in some cases beliefs should be maintained in an optimal, rational way. In other cases belief maintenance should result in beliefs that reflect the, possible irrational, beliefs held by humans. The challenge is to develop a framework that supports the variety of ways in which beliefs can be maintained. In addition, for reasons of development, the framework should be able to support the processes required for the situational assessment task. Taken together, this results in the following choices for the framework.

Time

Because the agent should be able to display rational as well as biased behavior, beliefs are not deleted. By retaining all beliefs, it is possible to retrieve what was believed before. This enables the modeling of perfect memory and rational behavior, as well as the modeling of the influence of old beliefs on current reasoning. A practical consequence of this choice is the emergence of a large, possibly inconsistent belief base. Therefore, it is necessary to denote when what was believed. We propose to do this by adding an argument to the belief predicate denoting the time at which the belief is formed.

Time-stamped beliefs enable the reasoning over beliefs in time. Reasoning over beliefs formed at different time points can be used to to determine what is currently believed to be the case. This process implements belief maintenance, and can be done in a variety of ways supporting varied behavior. Time-stamped beliefs also enable reasoning about (values of) beliefs in time, which is required for the research task. For situational assessment it is important to reason about time in an *absolute* way. For example, from the time that passes between the formation of two beliefs about a vessel's position and their respective values, its speed can be deduced. So for situational assessment it is required to reason quantitatively about the time passed between various beliefs about events, e.g., for deducing a new belief that explains them.

Another way to reason over beliefs in time is doxastic logic in combination with modal temporal logic. Doxastic logic is a modal logic concerned with reasoning about beliefs. Doxastic logic typically uses 'Bx' to mean 'It is believed that x is the case', and treats 'B' as a modal operator. Using a modal temporal logic, which views time as a sequence of states, it is possible to reason about beliefs over time. However, the temporal operators embedded in modal temporal logics have a *relative* nature. For example, they denote that a certain belief is *always* the case, *until* or *after* another belief holds, or *sometime in the future*. In contrast, we require the reasoning over beliefs in an absolute way and use *belief* to denote a predicate logic expression that might incorporate variables and values. Recently researchers have focused on developing modal predicate temporal logics that do enable the reasoning about variables and values (Areces and ten Cate, 2006), but this line of research is not yet fully developed. Therefore, we decided to explicitly add a time stamp to beliefs to enable (varied) quantitative reasoning over beliefs over time within a temporal predicate logic setting (Galton, 2006).

Source

Beliefs do not only need to be maintained based on time aspects, another important aspect is their source. For example, when source S1 states at T1 that the speed of X is 20 knots, and at T2 that it is 15 knots, the former can be discarded because it is obsolete. However, when instead of source S1 source S2 stated at T2 that the speed was 15 knots, the former cannot be discarded that easily. Instead, the two pieces of information should be integrated. In this process various factors can play a role, e.g, the trust in the sources, or the time passed between T1 and T2.

In situational assessment, information is generally obtained from a wide variety of sources. Sources range from memory to reference works as books and maps, and from systems like radar and sonar to humans in a variety of roles. The agent should be able to distinguish and integrate the information gathered from these different sources. We propose to support this by denoting the source of every belief, which can be external as well as internal. This explicit reference to the source of a belief also enables the modeling of the influence this source may have on the acceptance of its information.

Certainty

When assessing a situation, humans often have to deal with uncertain information. This is especially true in military contexts. The cognitive agents should therefore be able to represent and reason about uncertain beliefs. To facilitate this it is proposed to attach an absolute, numerical certainty value to beliefs, ranging from 0 to 1. 1 denotes a total

certainty in the (value of) the belief, 0 denotes complete uncertainty, i.e., that the value of the belief in unknown.

In case the value of a belief's predicate is numerical, the certainty label denotes the dispersion of the value believed. To model this, it is required to denote for each belief predicate the amount with which the standard deviation of its value increases when its certainty decreases. For example, every 0.1 with which it is less certain that the speed of a contact is 20 knots adds an additional dispersion of 1 knot. This choice of implementing uncertainty enables computation over these uncertainties when forming new beliefs (see, e.g., Figure 3.6 on page 118).

Selected Approach

Based on our prerequisites to model belief maintenance for the situational assessment task and to support a variety of ways to maintain beliefs, the choice is made to add a time stamp, source label, and certainty value to the beliefs of an agent. These arguments should enable the modeling of a variety of belief maintenance techniques.

We do not prescribe which belief maintenance techniques should be modeled. In the papers embedded in this chapter we propose certain options, but these are not firm. It is the explicit idea that the agent modeler can use the belief arguments to maintain beliefs in a variety of ways. For example, when it is desired to model rational source integration, Dempster's rule of combination could be applied as belief maintenance technique. Because the existing belief maintenance techniques and models presented in the previous section all embed a fixed way to maintain beliefs, none of them is simply adopted. Another reason is that none of them supports the explicit (quantitative) reasoning over beliefs in time, which is required for modeling the situational assessment task.

3.1.3 Chapter Overview

This chapter presents four papers. The first paper (Section 3.2) introduces the developed belief framework incorporating time, source, and certainty stamped beliefs. The framework is evaluated for its suitability to model biased behavior by implementing a cognitive agent that replicates the biased behavior that supposedly underlies the historic case of the U.S.S. Vincennes. In 1988 the commander of the U.S.S. Vincennes decided to engage an incoming aircraft that turned out to be an Iranian Airliner. This tragedy is often quoted as example of faulty decision-making under stress (Klein, 1998).

The belief framework is developed with the situational assessment task in mind. An example of this task is the compilation of a tactical picture of surface contacts in naval anti-surface warfare as trained by the Royal Netherlands Navy in the Action Speed Tacti-

cal Trainer (see Section 1.4.3). This Tactical Picture Compilation Task (TPCT) entails the classification and identification of surface contacts, which is a situational assessment process. The second paper (Section 3.3) describes the development of two cognitive agents capable of executing the TPCT, as well as the evaluation of their behavior by military experts. For the development of these agents, first a formal task model was developed based on expert knowledge and the belief framework. Two agents were implemented to evaluate the ability of the framework to produce on-line *rational* as well as *biased* behavior. The first agent BOA-R produces rational behavior, the second agent BOA-B incorporates typical biases and produces biased behavior.

The third paper (Section 3.4) provides further details concerning the translation process from the formal task model of the TPCT to the two software agents. The two BOA agents are implemented in ACT-R 6.0, unlike the cognitive agent presented in Section 3.2 that is implemented in LeadsTo (Bosse et al., 2007). LeadsTo is a language with supporting software developed to simulate dynamic processes in terms of both qualitative and quantitative concepts. This software is not suited for implementing an agent performing a cognitive task in real time in a simulated environment.

Our decision to implement the two BOA agents in ACT-R was dictated by an objective of the Cognitive Modeling program, namely to 'investigate existing architectures for implementing the cognitive models into intelligent agents, and for linking these agents to simulation systems' (Section 1.4.1). With the fourth paper (Section 3.5) we further carry out this objective: in this paper we present and discuss the reimplementation of the formal TPCT model in Soar. This reimplementation enables us to compare ACT-R and Soar with respect to the developed task model and underlying belief framework.

Additional Remarks

Because the papers embedded in this chapter have been written over a period of time, minor discrepancies exist in the notations used. To be precise, the first two papers denote a belief by *belief(p, a, v, t, s, c)*, while the last two write *belief(P(A, V), T, S, C)*. However, both denote using variables that it is believed by the agent at time T, based on source S and with certainty C, that predicate P holds for attribute A with value V.

In addition, Section 3.2 is the only section that actively denotes that a belief b is held in memory by the two-place predicate holds_at: *holds_at(b, t)*. The other sections simply assume this. Moreover, the first section of this chapter gives a new label to specific types of beliefs, e.g., *lastbelief*. In the task model presented in the other sections these types are represented implicitly by specific conventions. For example, a last belief is denoted by a time variable ending at a 1 (e.g., T1), a second-last by a 2 (e.g., T2), and a random belief before a relative one by ' (e.g., T2').

Research Paper

3.2 A Belief Framework for Modeling Cognitive Agents

Abstract

Simulation-based training in complex decision-making can be made more effective by using intelligent software agents to play key roles. For successful use in training, these agents should show representative behavior. Representative behavior may reflect expert behavior, but may also be far from optimal, especially under stress conditions. Current agent architectures hardly offer support to model cognitive properties that are essential to human decision-making. The present paper describes a framework in which agent's beliefs are extended with additional arguments with which such dynamic cognitive properties can be formalized. An historic military event is used to demonstrate that the resulting framework is capable of modeling representative behavior.

This section is published as:

Heuvelink, A. A Belief Framework for Modeling Cognitive Agents. In R. L. Lewis, T. A. Polk, and J. E. Laird (Eds.), *Proceedings of the 8th International Conference on Cognitive Modeling* (ICCM 2007), Psychology Press, p. 235-240. July 26-29 2007, Ann Arbor - Michigan.

3.2.1 Introduction

Organizations that operate in highly uncertain and dynamic environments, such as the military, require competent staff personal. However, the very nature of their missions makes it hard to setup real-world training. Scenario-based simulator training is considered an appropriate approach for training decision-making in complex environments (Oser, 1999). A main requirement of simulator training is that it correctly represents these aspects of the real world that are necessary to achieve the learning objectives. Perhaps the most important aspect of human decision-making is the interaction with other humans, e.g., team members. In order for simulation-based training to be an alternative of real-world training, simulated entities must be able to respond naturally and validly to any emerging situation. Therefore our goal is to develop agents that are capable of generating behavior that is representative for the human they represent.

There is growing conviction and evidence that we can develop such agents by capturing the human cognitive processes in a cognitive model. The research fields of Artificial Intelligence (AI) and Cognitive Science (CS) have yielded various architectures that can be used to develop cognitive models (Pew and Mavor, 1998).

We start this paper with describing properties of architectures that are currently used for cognitive modeling. We then elaborate on various typical features of human cognition and argue that these architectures lack the mechanisms to model these features. Next, we explain how more human-like behavior can be achieved by formalized reasoning rules on beliefs with additional arguments. We illustrate the strength of this belief framework by implementing a cognitive model of a key player in a historic military incident. Finally we draw conclusions on the significance of our work and propose future research.

3.2.2 Related Research

The potential and benefits of representing human behavior in (training) simulations by cognitive models of key players is generally recognized. As a result several models have been developed that can play such key roles (see, e.g., Gluck and Pew, 2005). In general these models are either implemented in a cognitive architecture, like ACT-R or SOAR, or in an agent architecture such as JACK or JADEX. Cognitive architectures embody a theory of cognition, while agent architectures often encapsulate Beliefs, Desires and Intentions (BDI) (Georgeff and Lansky, 1987). For all architectures it holds that they themselves are not a model, but that they offer the constructs to build a model. The most basic construct is a declarative information entity with which the knowledge of the agent can be represented. We will refer to these knowledge entities as *beliefs*.

Since it is important for an agent to have a correct and consistent view of the world, a

central issue is how an agent keeps a consistent belief database upon receiving information that is inconsistent with its current beliefs. Within AI the problem is generally solved by throwing away beliefs that cause the inconsistency. By doing so, the agent no longer has access to what it believed before, which is not very human-like. Cognitive architectures tend not to eliminate inconsistent beliefs but deal with them when they retrieve beliefs into working memory (e.g., Anderson and Lebiere, 1998; Paglieri, 2004). Mechanisms differ between architectures, but often the way they revise and retrieve beliefs is fixed. This aspect restrains the agent from having access to his old, possibly currently disbelieved, beliefs.

Research in cognitive science shows that the nature of the beliefs that form the inconsistency influence the way humans solve the inconsistency. For example, the *time* on which information is received has a large influence on the belief formation of humans. Famous temporal order effects in the updating of beliefs are primacy and recency (e.g., Anderson, 1981), which are considered to be typical human biases. Dieussaert et al. (2000) found that when a belief is deduced from a conditional statement (e.g., if A then B), that then upon receiving a categorical statement (not B) the initial *strength* of the belief in the conditional (A) is important for the revision of belief B. Another major finding is that the *source* of the beliefs is very important for how they are treated. Humans are biased to believe information that is obtained by one's own over information communicated by others. The trust of a human in the source of the information is another important factor for its believability (e.g., Mercier and der Henst, 2005).

Many more cognitive biases are found in the formation of and reasoning on beliefs (Wickens and Flach, 1988). The availability bias denotes the tendency of humans to focus on the most salient outcome, which is *time* related. The confirmation bias functions on two levels; it denotes the tendency to only search for information that confirms the current hypothesis as well as the tendency to give congruent information more weight than incongruent information. The latter strongly influences the *strengths* of beliefs. The as-if bias denotes the tendency of humans to treat *sources* 'as if' they are the same.

Cognitive biases influence the quality of human decision-making and are found to arise especially under stress conditions (see e.g., Baron, 2000). Since we want our agent to generate representative human behavior under a variety of stress conditions, we need to be able to model the before mentioned processes. Current architectures don't offer support to model (biased) reasoning over beliefs taking their initial time, their source and their certainty into account. In the next section we propose a framework with which such processes can be formalized.

We are not the first to tackle the problem of modeling biased reasoning and belief revision. However, as mentioned above most cognitive belief models adept the strengths

of beliefs upon receiving new information and by doing so loose access to what was believed before. Moreover, stress is often not a factor in the revision or retrieval of beliefs.

3.2.3 Belief Framework

We want to develop a decision-making agent that reflects a human in the way it acts and reasons. For this goal we develop a logical framework in which beliefs represent the agent's declarative information entities. We decide to represent beliefs in predicate logic since this format enables formal verification of global properties which is useful for training. To ensure that an agent can have an up-to-date consistent belief set without loosing access to its old beliefs, we propose to *time stamp* each belief at the time it is formed with that time. With this feature it is possible to model (biased) reasoning over beliefs over time. We found only one other paper that proposes to time-stamp beliefs. Sripada (1993) took this approach in search of a more efficient belief revision technique, but only looked at binary beliefs.

We on the other hand want our agent to have graded beliefs like a human and therefore further propose to *certainty stamp* beliefs. The certainty stamp of a belief denotes the strength of the agent's belief in its truth value at the time captured in the time stamp. Last we propose to *source stamp* each belief, by which the origin of the information is captured. Using these three extra arguments various cognitive processes can be formalized as will be shown in the next sections.

Belief Predicate

A belief can be seen as a collection of properties that can be captured with the following $belief$ predicate:

$$\begin{aligned}
&\forall p\, \forall a\, \forall v\, \forall t\, \forall s\, \forall c\, [\, belief(p, a, v, t, s, c) \;\leftrightarrow \\
&\qquad \exists b\, \exists e\, [\, beliefhasterm(b, e) \wedge \\
&\qquad\qquad termhaspredicate(e, p) \wedge \\
&\qquad\qquad termhasattribute(e, a) \wedge \\
&\qquad\qquad termhasvalue(e, v) \wedge \\
&\qquad\qquad beliefhastimestamp(b, t) \wedge \\
&\qquad\qquad beliefhassource(b, s) \wedge \\
&\qquad\qquad beliefhascertainty(b, c)\,]\,]
\end{aligned}$$

The core of a belief is a term that denotes the information that is believed, e.g., that the identity (p) of track2 (a) is hostile (v). Besides this term a belief consists of three extra arguments denoting the time it was formed (t), its source (s) and how certain the agent is of that belief (c).

To formalize relations between beliefs over time it is necessary to have a reference to time that specifies the time at which a certain belief was held by the agent. For this we introduce a two-place predicate HoldsAt. When we reify the belief predicate of the object language to a propositional term b, we can state using this meta-language predicate at which time the belief is held: HoldsAt(b, t).

For every belief(p,a,v,t,s,c) that can be found in the agent's database it can stated that HoldsAt(belief(p,a,v,t,s,c),t), since the t of the belief denotes that it was then formed and thus logically holds.

Formation of Beliefs

By using the aforementioned belief system we can model relevant cognitive properties and processes. The first interesting process is the transfer of information from the outside world into a belief. Research in cognitive science mentioned above pointed out that the source of the information as well as the current state of beliefs (confirmation bias) is relevant for this process. These two aspects influence the strength with which an agent ends up believing that information, i.e., the certainty of its belief. We accommodate these aspects by transferring information from the world into a belief in three stages.

First, a $presourceexpectancybelief$ is formed:

$$\begin{aligned}
&\forall p\ \forall a\ \forall v\ \forall t\ \forall s\ \forall c\ [\\
&\quad HoldsAt(input_from_world(p,a,v,s,c),t)\\
&\quad \rightarrow\\
&\quad HoldsAt(presourceexpectancybelief(p,a,v,t,s,c),t)\ ]
\end{aligned}$$

Secondly, the influence of the source on the believability of the given information is determined, by using the agent's trust level in that source. In how far this bias occurs, i.e., how much this process moves the perceived certainty away from the actual certainty, is influenced by the current stress level of the agent.

$$\begin{aligned}
&\forall p\ \forall a\ \forall v\ \forall t\ \forall s\ \forall c\ \forall tr\ \forall st\ [\\
&\quad HoldsAt(presourceexpectancybelief(p,a,v,t,s,c),t) \wedge\\
&\quad HoldsAt(trust_in_source(s,tr),t) \wedge && (-1 \leq tr \leq 1)\\
&\quad HoldsAt(stress(st),t) && (0 \leq st \leq 1)\\
&\quad \rightarrow\\
&\quad HoldsAt(preexpectancybelief(p,a,v,t,s,c+tr\times c\times st),t)\ ]
\end{aligned}$$

Thirdly, the current state of beliefs is taken into account. This is not done directly, but through the notion of expectancies. The $expectancy$ predicate has 4 arguments, denoting the expected term (p, a, v) as well as a certainty. Expectancies differ from beliefs in that they are formed automatically and can be considered unconscious.

Expectancies are formed in two ways; each term that is currently believed gets transferred to an expectancy that will hold the next time step. The strength of the expectancy is a function of the strength of the belief and the persistence of the predicate; we will elaborate on the latter later on. Secondly, certain (combinations of) beliefs can yield new expectancies. The certainty of expectancies decays over time and the expectancy ceases to exist when its certainty becomes equal to zero.

To determine the final certainty of the belief existing congruent and incongruent expectancies are taken into account. The extent to which these expectancies bias the certainty of the agent in the final belief is influenced by the current stress level. Since multiple situations are possible multiple rules are needed to formalize this process:

$$\begin{aligned}
&\forall p\, \forall a\, \forall v\, \forall t\, \forall s\, \forall c\, [\\
&\quad HoldsAt(preexpectancybelief(p, a, v, t, s, c), t) \wedge \\
&\quad \neg \exists w\, \exists d\, [\, HoldsAt(expectancy(p, a, w, d), t)\,] \\
&\quad \rightarrow \\
&\quad HoldsAt(belief(p, a, v, t + 1, s, c), t + 1)\,]
\end{aligned}$$

$$\begin{aligned}
&\forall p\, \forall a\, \forall v\, \forall t\, \forall s\, \forall c\, \forall d\, \forall st\, [\\
&\quad HoldsAt(preexpectancybelief(p, a, v, t, s, c), t) \wedge \\
&\quad HoldsAt(expectancy(p, a, v, d), t) \wedge \\
&\quad HoldsAt(stress(st), t) \\
&\quad \rightarrow \\
&\quad HoldsAt(belief(p, a, v, t + 1, s, c + d \times st), t + 1)\,]
\end{aligned}$$

$$\begin{aligned}
&\forall p\, \forall a\, \forall v\, \forall t\, \forall s\, \forall c\, \forall u\, \forall d\, \forall st\, [\\
&\quad HoldsAt(preexpectancybelief(p, a, v, t, s, c), t) \wedge \\
&\quad HoldsAt(expectancy(p, a, u, d), t)\ \wedge\ u \neq v \wedge \\
&\quad \neg \exists e\, [\, HoldsAt(expectancy(p, a, v, e), t)\,] \wedge \\
&\quad HoldsAt(stress(st), t) \\
&\quad \rightarrow \\
&\quad HoldsAt(belief(p, a, v, t + 1, s, c - d \times st), t + 1)\,]
\end{aligned}$$

Intermediate rules (not denoted) handle new (pre)beliefs whose certainties lie outside the certainty range. An agent can also form new beliefs using *conditional statements* and its current beliefs. These rules, together with believed *categorical statements*, make up the task specific knowledge of an agent. The formation of a new belief by a conditional statement happens in two stages. First a $preexpectancybelief$ is formed, which is than transferred into a belief using the mechanisms described above. A belief formed by a reasoning rule receives that rule's name as it source. An example rule is the following:

$$\forall c\,[\; HoldsAt(belief(weather, local, raining, t, integratedsources, c), t)$$
$$\rightarrow$$
$$HoldsAt(preexpectancybelief(status, street, wet, t, deduce\_wet\_from\_raining, c), t)\;]$$

Note that this rule requests as input a just formed belief (denoted by t) whose source is equal to $integratedsources$.

Belief Integration

An important aspect of the belief framework is that reasoning rules request beliefs as input that have as time argument the $current\ time$ and as source argument $integrated\ sources$. The requested time argument denotes the claim that the belief should just be formed and thus holds (present in working memory) while the source denotes the claim by which rule it should be formed. The reasoning rule that produces beliefs with $integratedsources$ as source argument deduces what exactly is currently believed by the agent. This rule deals with any inconsistencies in the belief set formed by beliefs from different sources or at different times. The retrieval of a belief into working memory can be seen as its human equivalent.

To implement this process we first implement the agent's memory by the following simple rule, which assumes that beliefs are never forgotten.

$$\forall p\,\forall a\,\forall v\,\forall t'\,\forall s\,\forall c\,\forall t\,[$$
$$HoldsAt(belief(p, a, v, t', s, c), t)$$
$$\rightarrow$$
$$HoldsAt(belief(p, a, v, t', s, c), t + 1)\;]$$

To facilitate the formalization of reasoning rules that use the agent's memory we introduce the $lastbelief$ predicate, which denotes the most recent belief in the agent's memory for given specifications. Its definition is:

$$\forall p\,\forall a\,\forall v\,\forall t\,\forall s\,\forall c\,\forall n\,[$$
$$HoldsAt(lastbelief(p, a, v, t, s, c), n)$$
$$\leftrightarrow$$
$$[\; HoldsAt(belief(p, a, v, t, s, c), t)\ \wedge\ t \leq n\ \wedge$$
$$\neg\exists t'\,[\; HoldsAt(belief(p, a, v, t', s, c), t')\ \wedge\ t' \geq t\ \wedge\ t' \leq n\;]\;]\;]$$

To determine what exactly is believed by the agent, it is relevant to consider that a belief's validity over time is strongly influenced by its predicate. Values of certain predicates are much more persistent than others; consider the chance that a person's sex, marital status or mood changes over time. An agent's certainty level in a belief whose predicate is very persistent does not change much over time. However, beliefs about

predicates of which the values are likely to change will quickly loose certainty. The persistence level of a predicate also influences the decaying factor of expectancies about it. The rule that determines what exactly is believed, so that is responsible of deducing the current belief from old beliefs, is formalized as:

$$
\begin{aligned}
&given(p, a)\\
&\forall v1\ \forall t1\ \forall s1\ \forall c1\ \forall t\ \forall pd\ \forall c'\ [\\
&\quad HoldsAt(lastbelief(p, a, v1, t1, s1, c1), t) \wedge\\
&\quad HoldsAt(persistence\_decay(p, pd), t) \wedge \qquad (0 \leq pd \leq 1)\\
&\quad \neg\exists c''\ \exists v2\ \exists t2\ \exists s2\ \exists c2\\
&\quad [\ HoldsAt(lastbelief(p, a, v2, t2, s2, c2) \wedge\\
&\quad\ c2 - pd \times (t - t2) > c1 - pd \times (t - t1)\ ]\\
&\quad \rightarrow\\
&\quad HoldsAt(belief(p, a, v1, t + 1, integratedsources, c1 - pd \times (t - t1)), t + 1) \wedge\\
&\quad \ldots\\
&\quad HoldsAt(belief(p, a, v1, t + 1, integratedsources, c1 - pd \times (t - t1)), t + 10)\ ]
\end{aligned}
$$

Also in this case there is an intermediate rule that handles beliefs whose certainties lie outside the certainty range.

Following this rule, the agent ends up believing the value of the belief whose certainty is the greatest after taking into account the time passed since it was formed and the persistence of the predicate. This might entail that an older belief with a higher certainty is believed over a newer belief from a different source or the other way around, it depends on the nature of predicate. The determination of the new certainty is currently kept straightforward; it is equal to the highest one after taking the time into account. Other sources that claim the same do not contribute to its certainty.

A belief that is consciously deduced using this rule is stated to hold for ten following time points. This reflects the fact that items retrieved by humans also stay a while in working memory. The above rule takes many aspects into account and is cognitive expensive. As mentioned on page 77 humans display a bias to treat all sources as equally likely. With this simplification a decision can be made much cheaper, for example, by simply taking the most recent one. In such cases the antecedent becomes:

$$
\begin{aligned}
&HoldsAt(lastbelief(p, a, v1, t1, s1, c1), t) \wedge\\
&HoldsAt(persistence\_decay(p, pd), t) \wedge \qquad (0 \leq pd \leq 1)\\
&\neg\exists v2\ \exists t2\ \exists s2\ \exists c2\\
&[\ HoldsAt(lastbelief(p, a, v2, t2, s2, c2)\ \wedge\ t2 > t1\ ]
\end{aligned}
$$

Which rule is applied is influenced by the agent's stress level and should be determined at the control level.

Reasoning over Beliefs over Time

With the given belief predicate we can deduce whether an agent believes something for a longer period of time. The *timecertaintyintegratedbelief* predicate denotes the time when the term of the current integratedsources-belief was believed for the first time. Furthermore it should hold that no other value was believed in the mean time and that it did not become unknown caused by the time passed and the decay of certainty:

$$
\begin{aligned}
&given(p, a, pd)\\
&\forall n\, \forall c\, \forall t\, \forall d\, [\\
&\quad HoldsAt(belief(p, a, v, n, integratedsources, c), n) \wedge\\
&\quad HoldsAt(belief(p, a, v, t, integratedsources, d), t) \wedge\\
&\quad \forall v'\, \forall t'\, \forall c'\, [\\
&\qquad HoldsAt(belief(p, a, v', t', integratedsources, c'), t') \wedge\\
&\qquad v \neq v' \ \wedge\ t' < n \ \wedge\ t > t' \wedge\\
&\qquad \neg\exists t''\, \exists e\, [\\
&\qquad\quad HoldsAt(belief(p, a, v, t'', integratedsources, e), t'') \wedge\\
&\qquad\quad t'' > t' \ \wedge\ t'' < t\,]\,]\\
&\quad \forall t'\, \forall c'\, [\\
&\qquad HoldsAt(belief(p, a, v, t', integratedsources, c'), t') \wedge\\
&\qquad t \neq t' \ \wedge\ t' < n \wedge\\
&\qquad \neg\exists t''\, \exists e\, [\\
&\qquad\quad HoldsAt(belief(p, a, v, t'', integratedsources, e), t'') \wedge\\
&\qquad\quad t'' > t' \ \wedge\ c' - pd \times (t' - t'') > 0\,]\,]\\
&\quad \rightarrow\\
&\quad HoldsAt(timecertaintyintegratedbelief(p, a, v, t), n)\,]
\end{aligned}
$$

Note that this rule can be made executable by replacing the HoldsAt(b, tx) statements with HoldsAt(b, n), given that a memory system is in place. This should obviously hold for an implemented model, as presented in the next section.

This extra object predicate is useful for modeling the deduction of a belief based on the persistence of another, e.g., position stays equal → speed = 0. The predicate is also very useful to model the reasoning over belief patterns over time. E.g., to determine whether a ship zigzags the beliefs over time concerning its headings have to be integrated. The following rule depicts the principle, but should be filled with more domain specific knowledge.

$$
\begin{aligned}
&given(p, a, v1, v2)\\
&\forall t1\, \forall t2\, \forall t3\, \forall n\, [\\
&\quad HoldsAt(timecertaintyintegratedbelief(p, a, v1, t1), n) \wedge
\end{aligned}
$$

$$
\begin{aligned}
&HoldsAt(timecertaintyintegratedbelief(p,a,v2,t2),t1') \ \wedge \ t1' < t1 \wedge \\
&\neg\exists t1'' \ \exists v \ \exists t \ [\\
&\quad HoldsAt(timecertaintyintegratedbelief(p,a,v,t),t1'') \wedge \\
&\quad t1'' < t1 \ \wedge \ t1'' > t1' \ ] \wedge \\
&HoldsAt(timecertaintyintegratedbelief(p,a,v1,t3),t2') \ \wedge \ t2' < t2 \wedge \\
&\neg\exists t2'' \ \exists v \ \exists t \ [\\
&\quad HoldsAt(timecertaintyintegratedbelief(p,a,v,t),t2'') \wedge \\
&\quad t2'' < t2 \ \wedge \ t2'' > t2' \ ] \\
&\rightarrow \\
&HoldsAt(preexpectancybelief(pp,a,vp,n,this\_rule,c),n) \ ]
\end{aligned}
$$

3.2.4 Case Study - Iran Air Flight 655

To illustrate our approach we present an historic case for which we developed and implemented a cognitive model of a human decision maker. It concerns the Identification Designation Supervisor (IDS) aboard the combat information center of the **USS Vincennes** cruiser that in 1988 erroneously shot down an Iranian Airbus (Fogarty, 1988). This accident has been widely referred to as an example of faulty decision-making under stress (Klein, 1998). Using this case, we want to investigate whether our approach can be used to model the behavior of the IDS-officer.

Table 3.1 gives a short description of the sequence of most relevant events that led to the wrong identification of the airbus by the IDS, which contributed to it being shot down. This description mixes facts about the behavior of the IDS with assumptions about his reasoning. We deduced both from the formal investigation rapport (Fogarty, 1988).

Cognitive Model of the IDS

Our approach focuses on formalizing belief predicates and processes on beliefs with which we can model *how* humans process information. The formalization of *when* they do that has not been tackled. However, to simulate a cognitive model that demonstrates the former, an implementation of the latter is needed. To simulate human control we use a simple goal-directed reasoning strategy. For this strategy to work we abstracted the necessary in- and output of each rule, added the goal it contributes to, and specified what satisfies that goal. For the example rule on page 80 two of these constructs would be:

$$
\begin{aligned}
&input\_of\_rule\_goal(deduce\_wet\_from\_raining, determine\_status(street), \\
&\qquad\qquad belief\_tc(local, weather, raining, integratedsources)) \\
&satisfies\_goal(determine\_status(street), belief\_vtsc(status, street))
\end{aligned}
$$

Time	*Events*
10.47 AM	• The IDS is focused on an Iranian P-3. Since the P-3 belongs to hostile country Iran and is a patrol aircraft that can guide other aircraft on hostile missions, the IDS expects hostile aircrafts. • The radar reports a new track of interest (track2) at a range of 47nm and bearing 025, which corresponds to the runway of Iranian airport Bandar Abbas. The IDS observes the new track and based on the fact that the track's origin is an Iranian airport also used for military aircrafts, he believes it might be hostile. • In order to determine whether the track represents a commercial aircraft, the IDS checks the Bandar Abbas commercial airline departure times schedule. However, the time of departure and scheduled time differ too much to make the neutral identification. • In order to obtain more information the IDS sets its remote control indicator (RCI) challenge gate at the track, so it can pick up the track's Identify Friend or Foe (IFF) Mode, a system all planes are equipped with. Based on his hostile assumption he expects to receive mode II or mode III. • The IDS picks up the neutral IFF Mode III-6675. However, all aircrafts can emit Mode III and therefore this information is not conclusive for a neutral identification.
10.48 AM	• The IDS observes from its Large Screen Display (LSD) that track2 is locked on by the USS Sides, however does not react. When military aircrafts are locked on to, they tend to change behavior. Non-military aircrafts do not notice when they are locked-on and therefore is unchanged behavior an indicator of a neutral aircraft. However, the IDS keeps believing the track might be hostile.
10.50 AM	• The IDS sees a Mode II-1100 on its RCI-display. He expected this response from the last track he queried and simply assumes that the signal comes from that track. • Since the IDS knows that a Mode II-11XX block is used by Iranian F-14's he reports track2 as 'possible F-14'.

Table 3.1: Description of Events that led to the wrong identification of the airbus

Furthermore we added backwards-reasoning rules as:

$$
\begin{aligned}
&\forall g1\ \forall p1\ \forall a1\ \forall r1\ \forall p2\ \forall a2\ \forall s\ \forall r2\ \forall g2\ \forall t\ [\\
&\quad HoldsAt(goal(g1), t) \wedge\\
&\quad HoldsAt(satisfies\_goal(g1, belief\_vtsc(p1, a1)), t) \wedge\\
&\quad \neg\exists v\ \exists s\ \exists c\ [\ HoldsAt(belief(p1, a1, v, t, s, c), t)\ ] \wedge\\
&\quad HoldsAt(output\_of\_rule\_goal(r1, g1, belief\_tsc(p1, a1)), t) \wedge\\
&\quad HoldsAt(input\_of\_rule\_goal(r1, g1, belief\_vtc(p2, a2, s)), t) \wedge\\
&\quad HoldsAt(output\_of\_rule\_goal(r2, g2, belief\_vtc(p2, a2, s)), t) \wedge\\
&\quad \neg\exists v\ \exists c\ [\ HoldsAt(belief(p2, a, v, t, integratedsources, c), t)\ ]\\
&\quad \rightarrow\\
&\quad HoldsAt(goal(g2), t)\ ]
\end{aligned}
$$

The main goal of the IDS-officer is to identify each track in the environment as quickly as possible in terms of *hostile*, *neutral* or *friend*. From this main goal all other relevant sub goals are determined each time step by backtracking, using the agent's task knowledge as well as its current belief state.

The model is implemented using the LEADSTO language with which temporal dependencies between two state properties can be modeled and depicted graphically (Bosse et al., 2007). The modeled dynamic properties have the following executable format: Let α and β be state properties of the form 'conjunction of atoms or negations of atoms', and e, f, g, h non-negative real numbers. In the LEADSTO language $\alpha \rightarrow_{e,f,g,h} \beta$ means:

If state property α holds for a certain time interval with duration g
Then after some delay (between e and f) state property β
will hold for a certain time interval of length h

In the following figures traces are shown that visualize the IDS properties (on the vertical axes) over time (horizontal axes). Dark boxes on top of a line denote that the property $HoldsAt$ that time, light boxes below that it does not. In all traces the certainty and persistence decay parameters range from 0-10 instead as proposed in the text from 0-1. For displaying purposes the $integratedsources$ beliefs that hold for 10 timestamps are summed up in one predicate $belief\_t$.

We lack the space to show all the reasoning steps of the IDS model, so we focus on the events of bullet 2. Figure 3.1 shows a trace depicting that the IDS actively observes the altitude of the track from its screen (ownCROD) and forms a belief about its value. This trace shows how the IDS's trust in his CROD (0.8) given his stress level (0.5) influences the certainty of the final belief (7 instead of 5).

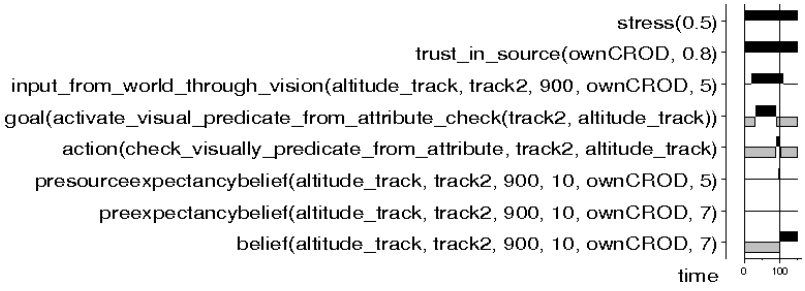


Figure 3.1: Observation of World and Formation of Belief

Next he reasons about the track's origin taking into account the track's position and altitude he just observed. The outcome, a belief about the airport it departed from, leads together with beliefs about the nature of that airport to a belief about the track's identity which is biased by the existing expectancy of hostile tracks (bullet 1), see Figure 3.2.

In the following time steps the IDS performs various actions that lead to new beliefs

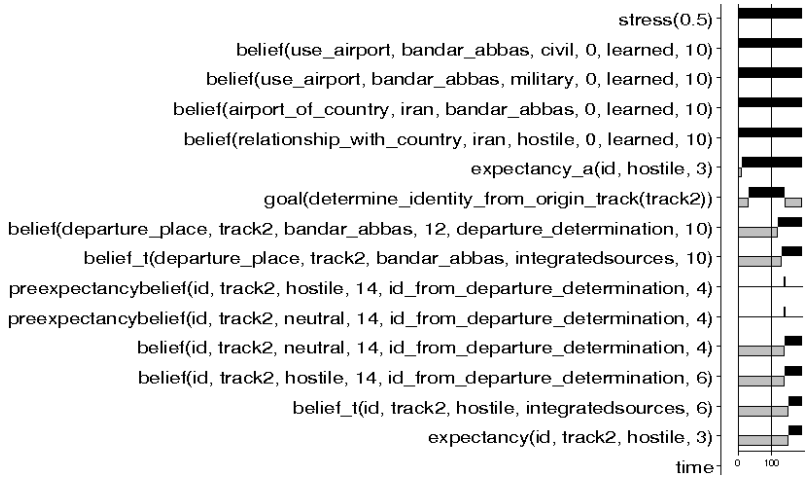


Figure 3.2: Formation of New Belief and Expectancy

that contribute to the reasoning about the track's identity. Unfortunately the IDS biased reasoning caused by his stress level causes him to belief he is dealing with a hostile F-16.

To illustrate one important aspect of our framework a bit further we made a trace that displays the source-integration process on two different types of belief predicates: see Figure 3.3. It can be seen that based on the nature of their predicate the beliefs are treated differently.

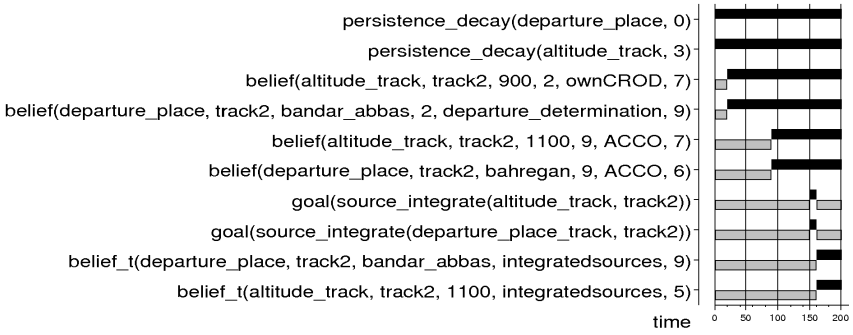


Figure 3.3: Source Integration on Two Predicate Types

3.2.5 Discussion and Conclusion

We developed a framework for cognitive modeling based on beliefs with a *time*, *source* and *certainty* label attached. These extra labels enable the formalization of various pro-

cesses on beliefs that lie at the basis of human cognition. Interactions between the time, source and certainty of beliefs has been made explicit, which is not possible in other architectures. Moreover, the influence of these parameters on each other is made tunable by the introduction of a stress level parameter.

The model of the IDS-officer shows that the framework is capable of generating human-like behavior. Agents modeled with this framework will be capable of generating more human-like behavior than, e.g., standard BDI agents. The fact that they are able to show behavior that is more representative for humans will make the agents more believable to the trainee that interacts with them. Since the believability of a training environment influences the effectiveness of the training, the modeling of agents using our framework will contribute to the effectiveness of the training and achievement of training objectives.

Our research does not stop here. The current framework will be extended by adding formal specifications of other relevant cognitive processes, such as attention and trust. Although the latter is already represented in the framework the current trust of an agent in sources is static. In reality however, trust is a dynamic property which is strongly influenced by experience. An agent capable of reasoning over its experiences with sources would be able to adapt its trust in sources. Stress level is another parameter that is currently fixed and that we would like to formalize as a dynamic property. Also the persistence values of properties are currently given and static, which is reasonable assuming that humans have learned them during their lifetime. However, when an agent would be capable to determine these values based on experiences with the environment, it would be much more adaptable to new environments.

As the next research step we will tackle the control of the agent. The simple control implemented in this paper was sufficient for demonstrating the reasoning rules. However, real humans have to deal with a limited amount of attention and processing power and therefore make many decisions on the control level. We like to develop a control framework in which we can capture these, probably biased, processes.

The cognitive validity of the model is debatable. However, by incorporating more outcomes of cognitive science research in our approach, we hope to approach our goal: the modeling of agents that can correctly represent human behavior in specific task training environments.

Acknowledgments

The author likes to thank Jan Treur for many fruitful discussions during this research and assisting on formal details, Tibor Bosse for clarifying aspects of the LEADSTO language and Karel van den Bosch for commenting an earlier draft.

Research Paper

3.3 BOA: A Cognitive Tactical Picture Compilation Agent

Abstract

Simulation-based training in complex decision-making can be made more effective by using intelligent software agents to play key roles, such as teammates, opponents and instructors. This paper presents a cognitive software agent that is capable of compiling a tactical picture in the domain of naval Anti-Surface Warfare. The agent is implemented in ACT-R and can perform this task in a simulated environment with a varying degree of quality, so it can play all the roles in a representative way. The agent's behavior was evaluated in a session with military experts and although much work remains to be done, the research was positively evaluated.

This section is published as:
Heuvelink, A., and Both, F. BOA: A Cognitive Tactical Picture Compilation Agent. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology* (IAT 2007), IEEE Computer Society Press, p. 175-181, November 2-5 2007, Silicon Valley - California.

3.3.1 Introduction

Military organizations tend to operate in highly uncertain and dynamic environments and therefore require competent commanders. However, the very nature of military missions makes it hard to setup real-world training. Scenario-based simulator training is considered an appropriate approach for training decision-making in complex environments (Oser, 1999). A main requirement of simulator training is that it correctly represents these aspects of the real world that are necessary to achieve the learning objectives.

Perhaps the most important aspect of tactical decision-making is the interaction with other humans, e.g., teammates or opponents. It is well known that the decision-making of humans in such contexts is structurally affected by biases (Fewell and Hazen, 2005). In order for simulation-based training to be an alternative for real-world training, simulated entities must be able to respond naturally and validly to any emerging situation. In light of the above this response may reflect expert behavior, but might also be far from optimal, especially under stress conditions.

In general, it is difficult to develop such autonomous virtual entities. Therefore, in training, Subject Matter Experts (SMEs) often control these entities. SMEs have the expertise to take the situational context into account and use this understanding to representatively play a role in the training scenario or evaluate (on-line) the appropriateness of trainee behavior. The benefit of using SMEs is that it is possible to deliver realistic tactical training scenarios. The main disadvantage of this approach is that the need for SMEs to deliver training elevates costs of training and requires high organizational and logistic efforts.

It would therefore be highly beneficial if we could develop software agents that are capable of generating representative and varied behavior like the SMEs can. There is growing conviction and evidence that we can develop such agents by capturing human cognitive processes in a cognitive model, and by implementing this model in a cognitive architecture (Pew and Mavor, 1998; Ritter et al., 2003).

This paper presents our research that has as goal the development of a cognitive agent capable of generating (online) representative behavior for a specific task in a simulated environment. The agent should in the future be usable for any of the introduced roles. For this reason it should be possible to adept the quality of its behavior to the role it plays. As task the compilation of a tactical picture is selected, which is a very important task in naval Anti-Surface Warfare (ASuW). The developed agent might in the future aid ASuW training by performing this task in a role as opponent or as own teammate. Its behavior might also be used for instruction by comparing the behavior of a trainee to it.

The paper starts with the introduction of the research domain and task. Then, by analyzing task and future role characteristics, various model requirements are discovered.

After elaborating on the cognitive model the translation of the model into ACT-R is described. For the evaluation of the resulting cognitive agent by SMEs a simulation environment is developed, which will be introduced and followed by a description of the performed evaluation. Finally the results of the evaluation are described and discussed.

3.3.2 Research Domain

The Royal Netherlands Navy (RNLN) supports research into the development of cognitive agents for training, stimulated by the potential of future integration with their training simulator, the Action Speed Tactical Trainer (ASTT). The ASTT represents a Command Center (CC) on a ship and can be used to train, among others, Command Central Officers (CCO).

The CCO aboard a ship is responsible for Anti-Surface Warfare (ASuW). For good performance the CCO has to have good situational awareness, which requires an up-to-date tactical picture of the situation. The currently described Tactical Picture Compilation Task (TPCT) entails the *classification* (e.g., is it a fisher, merchant or military ship) and *identification* (e.g., is it a neutral or hostile ship) of surface contacts.

The TPCT consists of the following subtasks:

- Observing direct classifiable behavior, e.g., speed of contacts or whether they fire.
- Matching observed behavior with information about the environment as well as intelligence, e.g., position relative to sea-lanes or expected threat direction.
- Monitoring all other behavior over time, e.g., sailing patterns or variations in speed.
- Classifying information using identification criteria (IDCRITS).

Besides the information that can be observed from the Large Screen Display (LSD) in the CC, the CCO can gain extra information and thus certainty about the identity of contacts by sending a helicopter for visual identification. Furthermore, it can be decided to change the Emission Control strategy (EMCON), which specifies the status of the own sensors. An active sensor can gain more information about the environment, but at the same time also emits information to the environment which can be undesired.

3.3.3 Cognitive Agent Requirements

The task characteristics introduced in the last section set some requirements for the cognitive agent. For correctly performing the TPCT the agent should at least be capable of:

- Observing information, e.g., a contact's speed.

In addition it should be capable of reasoning about the observed information, which can be divided in three subcapabilities derived from the last three subtasks. The agent should be capable of:

- Deducing new information by combining various pieces of information, e.g., whether a contact's position lies within a sea-lane.
- Deducing new information by integrating information over time, e.g., the maneuver a contact makes by integrating its positions.
- Classifying information, e.g., whether the known information is sufficient for identifying a contact as hostile.

Furthermore, the agent should be capable of performing actions that will actively yield new information. To be specific it should be capable of:

- Selecting the best action, e.g., where to send its helicopter to.

For this it should be capable of:

- Reasoning about the possible consequences of certain actions, e.g., which visual identification will decreases the uncertainty threat the most?

In order to generate behavior the agent should not only posses the capabilities to sense, reason, and act, it should also know when to bring which capability into action. For this it is required that the agent is equipped with a control function.

Additional cognitive agent requirements are set by the future applications of the agent. For fulfilling the role of a teammate or instructor it might need to explain its own behavior. To facilitate this future interaction between the cognitive agent and the trainee, it was decided to model the agent using Beliefs, Desires and Intentions (BDI) (Georgeff and Lansky, 1987). The BDI framework is well known within AI for modeling proactive, goal-directed agents. Since its terms are closely related to notions in Folk Psychology the explanation of the behavior of a BDI agent is intuitively understandable by humans.

Depending on its future application, the agent either needs to perform the task in a rational expert-like manner (e.g., generating expert behavior for comparison with trainee performance), or perform the task like a real person, including commonly made mistakes (e.g., generating biased behavior to train the trainee to deal with faulty teammates). In general there are two types of mistakes humans make: a) mistakes that result from unknown or false task knowledge and b) mistakes that result from inherent properties from human cognition. The latter type of mistakes may arise at the beginning of training when a task is new, requires much attention and is thus stressful. It may, however, also occur by well trained people, especially under stress conditions. For generating representative behavior it is therefore required to incorporate the mechanisms causing such mistakes in our agent.

3.3.4 Cognitive Model and Agent Development

Dam and Arciszewski (2002) elicited most of the task knowledge necessary for performing the TPCT from SMEs. The few knowledge gaps that were discovered were filled in by further expert consultation. As a result this research could focus on capturing the elicited task knowledge in a cognitive model that, when implemented, is capable of showing representative task performance that can vary in quality.

As mentioned before it is well known that human situation assessment and decision-making is structurally affected by bias (Fewell and Hazen, 2005; Perrin et al., 1993). For example, humans tend to regard an experienced event as typical for the whole population, even if they are told that the experienced event was untypical (*representativeness bias*). Another well-known bias is the tendency of humans to focus on cues that support their current hypothesis, no matter how tentatively held, and disregard contradictory cues (*confirmation bias*). Many other cognitive biases affecting the soundness of decision-making have been identified and are found to arise especially under stress conditions (Baron, 2000).

In meetings with instructors of the naval Operational School we identified common mistakes of trainees. Next, we coupled these mistakes to general biases that are likely to be their source. For example, trainees tend to focus solely on contacts that come from the expected threat direction, which is an expression of the confirmation bias. We want to incorporate the mechanisms that generate common mistakes in our cognitive model, so we can generate biased, student-like, as well as rational, expert-like, behavior. To model this variety in cognitive behavior we developed a framework that supports the modeling of (biased) reasoning over beliefs by taking their initial time, source and certainty into account. For details see (Heuvelink, 2007).

Beliefs form the basis of the logical framework and represent the agent's declarative information entities. Beliefs are denoted with the belief predicate: $belief(p, a, v, t, s, c)$. Beliefs have as their core a term that denotes the believed information, e.g., that the identity (predicate p) of contact x (attribute a) is hostile (value v). Besides this term a belief consists of three extra arguments denoting the time it was formed (t), its source (s) and how certain the agent is of that belief (c).

In addition to declarative knowledge that can be captured with the introduced belief predicate, the cognitive model also needs procedural knowledge. An agent capable of the TPCT needs rules that embody the six subcapabilities necessary for the task mentioned in the previous section, as well as a control function that determines when it should apply which rule.

The first subcapability - the observation of information and transformation of it into a belief - is general and task independent, see (Heuvelink, 2007). In the process of

transforming information into beliefs the certainty of the belief to be constructed can be biased by the trust of the agent in the source of the information. The other subcapabilities require and incorporate elicited task knowledge. The belief arguments introduced by the belief framework enable and facilitate reasoning. An example rule of subcapability 2 - the deduction of a new belief by combining various beliefs - is:

Determine_In_Sea_Lane_Contact(x)

$$\forall t\ \forall p\ \forall s\ \forall c\ \forall s1\ \forall cd\ \forall t1\ \forall r\ \forall c'\ [$$
$$[\ belief(position\_contact, x, p, t, s, c) \wedge$$
$$belief(sealane\_coordinates, sl, cd, t1, map, 1) \wedge$$
$$\textbf{Possible\_Positions}(p, c, [r]) \wedge$$
$$c' = (number\_of\_[r \in cd])\ /\ (number\_of\_[r]) \wedge$$
$$c' > 0\ \wedge\ time(t)\ ]$$
$$\rightarrow$$
$$[\ belief(in\_sea\_lane\_contact, x, sl, t+1, determine\_in\_sea\_lane\_contact, c')\ ]\ ]$$

This rule deduces whether the agent beliefs that a certain contact (x) is in a sea-lane. **Possible-Positions** is a function that uses a position value and a certainty of that value and provides a list with possible position values. Given this list it is determined how certain the agent beliefs the contact is positioned within a sea lane.

The modeling of rules embodying subcapability 3 - the deduction of a new belief by integrating beliefs over time - is facilitated by the time label of beliefs:

Determine_Coherent_Sea_Lane_Contact(x)

$$\forall t\ \forall l\ \forall s1\ \forall c1\ \forall t1sc\ \forall s2\ \forall c2\ \forall c3\ [$$
$$[\ time(t) \wedge$$
$$belief(in\_sea\_lane\_contact, x, l, t, s1, c1)\ \wedge\ c1 > 0 \wedge$$
$$belief(in\_sea\_lane\_contact, x, l, t1sc, s2, c2)\ \wedge\ c2 > 0 \wedge$$
$$c3 = (t - t1sc)\ /\ \text{uncertainty\_time\_coherent\_sea\_lane}\ ]$$
$$\rightarrow$$
$$[\ belief(coherent\_sea\_lane\_contact, x, l, t+1, determine\_coherent\_sea\_lane\_contact, c3)\ ]\ ]$$

This rule deduces whether the agent believes that a contact's positions over time are coherent to a sea-lane. With t1sc we denote the retrieval of the oldest belief on whether contact x is in sea-lane l after which the s2 and c2 may have changed but it kept holding that c2 $>$ 0. The variable *uncertainty-time-coherent-sea-lane* determines how quickly the agent becomes certain of that a contact is coherent to a sea-lane, based on the time it already beliefs it is in there. The confirmation bias may influence this value based on whether coherence is expected.

Subcapability 4 entails the reasoning process concerning a contact's classification and identification and is implemented using a Naive Bayesian Classifier. This mechanism forwards the certainty of relevant beliefs to the certainty of a contact's classification or

identification. This final certainty may be biased by the confirmation bias on the basis of existing beliefs.

As stated before, procedural knowledge consists besides (biased) reasoning rules also of knowledge about when to apply these rules. This control function is implemented in the cognitive model by the desires and intentions it can have. The main desire of the agent responsible for the TPCT is the correct and quick identification of all contacts. The agent can perform three activities that will help him in reaching its main desire; it can reason about contacts' information accessible through its LSD, it can reason about and decide to send the helicopter to identify certain contacts, and it can reason about and decide to change its EMCON. When one of these activities has the agent's attention we say it is the agent's current intention.

The goal of our research is the development of a cognitive agent capable of generating representative task behavior. In the current study we focused on the development of the belief framework with which we can model the (biased) reasoning rules of the agent. It was decided to only model a simple control function. The basic intention of our agent is reasoning about information that is accessible through its LCD and it performs this task for each contact by looping through a list of all known contacts. At regular times, after checking ten contacts, the agent's attention shifts and it becomes its intention to reason about sending the helicopter. When that task is finished it becomes its intention to check whether it should change EMCON, after which it returns to its basic intention.

It would be more realistic when the agent is triggered to start reasoning about sending the helicopter or changing EMCON by a change in the environment. However, that requires a constant checking of whether that trigger happened, i.e., parallelism. True parallelism is hard to implement, we therefore simulate it by switching between the tasks at a regular basis.

Translating the cognitive model to ACT-R

A cognitive agent is a cognitive model implemented in software. We named our agent BOA, which is the Dutch abbreviation of Picture Compilation Agent. We decided to implement our cognitive model in ACT-R (Anderson and Lebiere, 1998), which is a software architecture embodying a theory of cognition.

In this paper we will not go into great detail concerning the translation process from our model to ACT-R, for details see (Both and Heuvelink, 2007). In general it turned out that it is well possible to translate the model's declarative knowledge to ACT-R; each belief can be captured by a chunk in ACT-R's declarative memory. In addition the modeled control could be translated to reasoning rules about the agent's goals in ACT-R's procedural memory without much difficulty.

An example of such a control rule in ACT-R is:

```
(p select-next-contact-goal1
   =goal>
      ISA         commitment1
      goal        monitor-contacts
      state       next-contact
      contact     =contact1
==>
   !bind! =contact2 (determine-next-contact)
   !bind! =new-intention (= (mod *contact-counter*
                            *max-number-of-contacts*) 0)
   =goal>
      plan        read-basal-info
      state       start-step
      contact     =contact2
      new-intention  =new-intention )
```

The antecedent of the rule requires that the agent is currently committed to monitoring contacts and that the agent is going to select a new contact. In the consequent, the agent determines whether to select the next contact, or to start a new intention. The slot `new-intention` of the goal chunk is filled with either true or false, influencing the production rule that can fire.

In contrast with the above it turned out to be hard to translate the model's reasoning rules into ACT-R's production rules. The main reason for this is that typically in the antecedent of a reasoning rule of our model, multiple (arguments of) existing beliefs are compared to generate a new belief, i.e., the rule's consequent (see the examples at page 93). However, a main characteristic of ACT-R is that a maximum of one chunk, i.e., belief, can be retrieved from declarative memory, and reasoned upon in working memory. To work around this problem the multiple beliefs that are needed in a reasoning rule are serially retrieved and temporarily stored in the goal buffer. They are then reasoned upon in LISP, the programming language in which ACT-R is implemented.

3.3.5 Simulation Environment

In order to test whether BOA indeed displays representative cognitive behavior we need to couple it with an environment that it can observe and where it can perform actions in. For this purpose we have build a simulation environment resembling the ASTT in GameMaker (Overmars, 2009), see Figure 3.4.

The basis of the environment is a 2D representation consisting of land and water and tactical markers like territorial waters, sea-lanes (the 'high ways' of the sea, often connecting harbors) and furthest on circles (FOC, circle showing the possible locations of

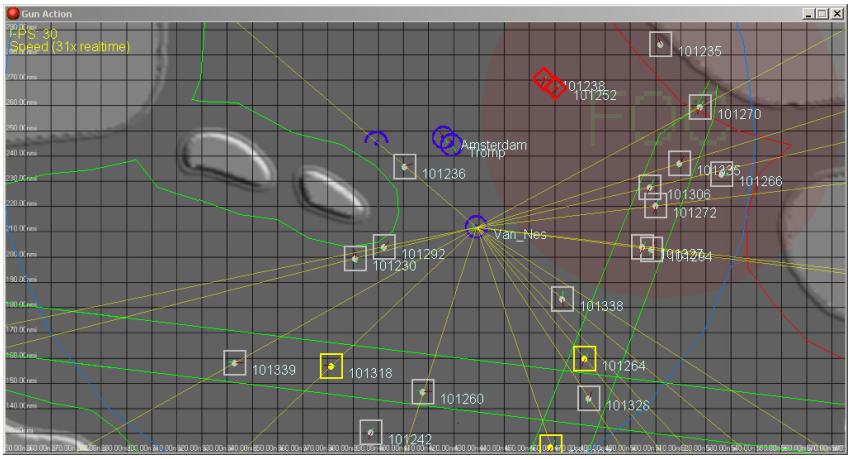


Figure 3.4: The Large Screen Display simulated in GameMaker

the enemy calculated from the last seen location and its maximum speed.) In the environment the following entities are modeled: ships, a helicopter, and a Marine Patrol Aircraft (MPA). These entities can carry the modeled systems passive and active radar. With passive radar other entities that use their active radar can be detected and it is possible to determine their approximate location and radar type. With active radar it is possible to detect any entity within radar range and to locate its position, speed, and heading. Furthermore, it is possible for friendly entities to share radar information.

Through a coupling between ACT-R and GameMaker, BOA can observe the information visible at the LSD, select a contact and observe its available information, and change its classification and identification. Furthermore, BOA can control the radar of the own ship and order its helicopter to visual investigate certain contacts.

Two windows show the viewer what is happening in the simulation. One window displays notifications from the modeled entities and systems, e.g., the loss of a radar contact or a visual identification report from the helicopter. The other window displays the radar contact currently selected by the agent with its available information, e.g., its speed and radar type. Assessments and/or assessment modifications made by the agent are also displayed in the second window.

3.3.6 Empirical Validation

Before we can use our agent in training applications we have to be sure its behavior is valid. In a study we assessed the face validity of our agent, being the subjective experienced similarity between the agent's behavior and the behavior of the human it

represents.

Our subjects were instructors of the naval Operational School. Their years of experience in training the TPCT ranged from 0.5 to 4 (avg. 2.6), their years of practicing the task from 2 to 10 (avg. 5).

We developed three scenarios for the task environment simulated in GM. In each scenario a friendly MPA was present flying a predetermined route while building up an image of the environment with its active radar, and sharing this information through the LSD. Using two of these scenarios we made four films, depicting either the behavior of the rational agent (BOA-R) or that of the agent with biases (BOA-B) during the course of a scenario. We made BOA perform the task in 5 times real-time, but due to the length of the scenarios (approximately 7.5 and 10.5 hours) there were long void periods in between assessments. To make matters practical the play back speed of the film was increased 4 more times, creating two films of 22 and two of 31 minutes. Next, we extended the films with balloons that elaborated on the agent's reasons for its performed actions.

We developed three types of questionnaires for evaluating respectively the simulation environment, the agent, and the scenarios. The questionnaires with which we wanted to evaluate the behavior of BOA were tailored to the behavior of the agent seen on film. The general structure of the questions was:

- Boa did so and so: do you consider it plausible a student would have made that decision?
- Would you as an expert have made that decision and if not, why do you think a student would have done so?
- Would you as an expert have made any other decision during the last period?

The questions were to be answered with yes or no together with an explanation, for which we provided blank space. At the end we also asked them to grade BOA with a mark between 0 and 10. The two other questionnaires also consisted of open yes / no questions and of questions in which the subjects were asked to grade certain aspects from 0 to 10.

The experiment was set up as a 2x2 design assuming the participation of eight subjects. Each subject viewed two films, one depicting BOA-R and one BOA-B, and one displaying scenario 1 and one scenario 2. To correct for order effects half of the subjects viewed the rational agent before the biased agent; the other half viewed the biased agent before the rational one. The same procedure was followed for the scenario order.

Our experiment started with introducing the subjects to the simulation environment in GM by asking them to perform the TPCT themselves in the third developed scenario. After completing the scenario we administered the questionnaire about the environment. Next, we played the two films that would stop after approximately every five decisions

made by BOA concerning the classification or identification of contacts or the deployment of the helicopter or radar. At every break we asked our subjects to fill in the questions of the questionnaire about the behavior of the agent. At the end we administered the third questionnaire in which we asked the subjects about the quality of the scenarios and the research in general.

3.3.7 Results and Discussion

Six participants completed the experiment.

Grade for:	**Grade**	**Standard Deviation**
Quality Simulated Environment	7	1.3
Quality Scenario 1	7.3	2.1
Behavior BOA-R in Scenario 1	5.7	1.5
Behavior BOA-B in Scenario 1	6	2.0
Quality Scenario 2	7.3	1.2
Behavior BOA-R in Scenario 2	5.3	2.1
Behavior BOA-B in Scenario 2	6.7	1.5
Usefulness Research Project	8.3	1.0

Table 3.2: Graded expert opinions

The grade given to the simulated task environment by the subjects (7), as well as their answers to the open questions, indicate that the developed environment can be used for training the TPCT. On the other hand, subjects mentioned some aspects that could be improved. The suggested improvements are mainly the addition of functionalities to the MPA and helicopter, which would increase the validity of the task.

The subjects graded both scenarios as suitable for training the TPCT (2 x 7.3). Furthermore, they stated that the scenarios and the behavior of the modeled entities were realistic enough, but that the scenarios should be made more difficult for advanced training.

The most interesting result is that the BOA in which we implemented the mechanisms that can lead to typical mistakes, i.e., biases, outperforms the rational BOA (6 and 6.7 versus 5.7 and 5.3 in the respective scenarios). The answers to the open questions indicate that the subjects in general agreed with the classifications and identifications that both the BOAs gave to contacts. However, they did not agree with the timing and order of these decisions: they often considered the BOAs to be too slow.

A combination of two factors causes BOA to react slower to changes in the environment than humans would. First, the translation of the cognitive model into ACT-R turns out to be slow. It takes BOA several seconds to reason about an identification or about

sending the helicopter. Second, BOA has a linearly control structure. When it is its intention to check and reason about information of contacts it does so by looping through a random list of known contacts. The consequence of these factors is that it simply takes time for BOA to reach a contact in the list, to start reasoning about it and to notice any relevant changes. And when BOA is slow in noticing relevant changes, it is also (too) slow in making decisions based on these changes.

A second consequence of the random order of contacts in the list is that BOA concordantly turns its attention to contacts at random. This behavior is very different from humans whose attention is influenced by, e.g., the closeness and importance of contacts. Moreover, humans treat members of a group simultaneously and do not alternate them with others like BOA did, and which was considered as unnatural.

BOA's decisions concerning the deployment of its helicopter were received with mixed feelings. In general the subjects would have sent the helicopter airborne much sooner, and most of them would not have let it return to the ship. This indicates that the threshold that determines whether the helicopter should go or stay airborne given a certain threat was modeled with a wrong value. However, the subjects did agree with BOA's choice of helicopter targets, which it makes by assessing the threat level of a (group of) contact(s) by their distance and current identification.

The biased version of BOA deployed the helicopter quicker than the rational BOA. This was probably the reason why subjects favored the biased version over the rational one. The reason that BOA-B was quicker than BOA-R lies within the implemented bias-mechanisms. Tactical knowledge was available in both the scenarios (e.g., knowledge concerning the possible position of the hostile contacts), which caused that some contacts (e.g., within the FOC) were already slightly believed to be hostile by BOA. The confirmation bias present in BOA-B causes it to give more weight to information that confirms a current hypothesis than to information that disconfirms it. This resulted in that given some other suspicious behavior, BOA-B quickly considered these contacts as very suspicious, and therefore as posing enough threat to send its helicopter airborne for visual identification. BOA-R on the other hand was less sure of their hostility, and did not consider it necessary to send the helicopter.

An interesting question is whether after decreasing the helicopter-threat-threshold, BOA-B will still be judged better. It might be that the confirmation bias makes BOA-B jump to conclusions about identifications and therefore too quick in selecting helicopter targets.

At just one moment in the scenarios it was relevant for BOA to make a decision about its radar status. The subjects' judgment varied; some judged its behavior as valid, others would have done something different.

The highest mark (8.3) was given to the usefulness of this kind of research for the navy. The subjects consider the future possibility of replacing (part of the) human agents by software agents as very beneficial for training. They were most enthusiastic about the idea of on-board training requiring nothing but a laptop.

3.3.8 Conclusion and Further Research

We build an agent in ACT-R that is capable of performing the tactical picture compilation task in a simulated task environment in a rational or biased manner, and evaluated its performance in a pilot study with naval experts. The study made clear that for future use in training applications, or in the related field of decision-support, an agent should show behavior that represents an expert or trainee better than BOA-R and BOA-B respectively currently do. Part of this is likely to be achieved by fine-tuning model parameters and by implementing the cognitive model in other, faster, software. Furthermore, the study showed that much can be gained by the implementation of valid, cognitive control.

Our next research step therefore consists of the development and implementation of a cognitive control framework. With such a framework it is not only possible to model biases that take place on the level on individual rules, but also on the level of control, e.g., which contact or task receives attention. When the rational agent will perform the tactical picture compilation task at the level of an expert, it will be possible to research how suited the mechanisms implemented in the biased agent are for generating representative common mistakes.

Overall, the naval experts were enthusiastic about BOA and could imagine such a cognitive agent aiding the training of the tactical picture compilation task in the future. With this research we have set a perhaps small, but certainly a definite step into the future in which training can take place at any time and at any place thanks to always available, capable, and willing role players.

Acknowledgments

The authors like to thank the instructors of the Opschool, especially LTZ 2OC Daan Smit, for their cooperation with this research, Willem van Doesburg for developing an initial version of the training environment and Karel van der Bosch for coaching this project as well as commenting an earlier draft.

Research Paper

3.4 From a Formal Cognitive Task Model to an Implemented ACT-R Model

Abstract

In order to generate behavior that can be validated, a cognitive task model needs to be implemented in software. This paper first introduces a cognitive task model based on a BDI framework and then focuses on the translation of that model into the ACT-R theory and software. Problems encountered during this translation process are described and further implications are discussed. It is found that the model's control structure matches well with ACT-R, but that the translation of its reasoning rules is complex. ACT-R's theory of cognition does not match with properties of our task model on three issues: 1) the number of memory items that can be stored in working memory, 2) the way memory items are retrieved from long-term memory, 3) the ability to execute complex computations.

This section is published as:
Both, F., and Heuvelink, A. From a Formal Cognitive Task Model to an Implemented ACT-R Model. In R. L. Lewis, T. A. Polk, and J. E. Laird (Eds.), *Proceedings of the 8th International Conference on Cognitive Modeling* (ICCM 2007), Psychology Press, p. 199-204. July 26-29 2007, Ann Arbor - Michigan.

3.4.1 Introduction

The development and implementation of cognitive task models in order to create agents that can replace humans for performing certain tasks in certain environments is a promising research activity (Gluck and Pew, 2005). Two important activities in the development of a cognitive model are the modeling of relevant task knowledge and the modeling of a cognitive valid theory of task execution. These modeling activities typically yield a formal design on paper of the cognitive task model. However, for studying the model's behavior and, to use the model as a replacement for a human in a certain environment it needs to be implemented in software.

The work presented in this paper is part of a greater research project that has as its goal the development of a cognitive agent that can perform the task of compiling a tactical picture on board of a ship in a training simulation. The current paper describes and reflects on the translation of the cognitive task model into the cognitive architecture ACT-R (Anderson and Lebiere, 1998). It will not go into details of the task, or into the validity of the behavior that the resulting cognitive agent shows.

First, we will briefly introduce the developed cognitive task model with its main properties concerning task knowledge and task control. Next, we discuss the translation of this model into ACT-R. To test the resulting cognitive agent we coupled it to an external simulation environment. This coupling and the general task performance of the ACT-R model are discussed. Furthermore, we reflect upon problems encountered during the translation as well as their implications. Finally, we lay down further research plans.

3.4.2 Research Domain

The task we modeled is the Tactical Picture Compilation Task (TPCT) within the domain of naval warfare. This task revolves around the classification and identification of entities in the surroundings. The warfare officer responsible for the TPCT monitors the radar screen for radar contacts and reasons with the available information in order to determine the type and intent of the contacts on the screen. The cognitive model of the TPCT is based on a Belief, Desire and Intention (BDI) framework (Georgeff and Lansky, 1987). This choice facilitates the translation of domain knowledge into the model since domain experts tend to talk about their knowledge in terms similar to beliefs, desires and intentions. The domain knowledge needed for performing the TPCT has been elicited from naval experts by van (Dam and Arciszewski, 2002).

The goal of our research is the development of cognitive agents that can be used for training purposes. We therefore want to model cognitive behavior, which can vary in level of rationality. To make this possible, we developed a specific belief frame-

work (Heuvelink, 2007). Three arguments are added to beliefs: a time stamp, the source of the information and a certainty level. Usually in BDI models, beliefs are thrown away as soon as a new belief is created that causes an inconsistency. However, because we want to reason over time, every belief is labeled with the time it is created and is never thrown away. The source and certainty labels make it possible to model an agent that can reason about information from multiple sources and with uncertainty, and that might do this in a biased way. A belief according to this new framework consists of a predicate P with an attribute A and value V, a time stamp T, source S and certainty C. An example belief is: $belief(identification(contact1, friendly), 12, determine\_id, 0.7)$

3.4.3 Cognitive Task Model

A cognitive task model typically consists of declarative knowledge, denoting facts, as well as procedural knowledge, denoting reasoning rules. Besides modeling how to reason, it is also necessary to model when to reason about what. In this paper, we mainly address the modeling of the declarative and procedural knowledge necessary for performing the TPCT. The modeling of cognitive control is not our current focus, and therefore we limit the complexity of our control structure. First, we describe the format of the declarative and procedural knowledge embedded in the cognitive model. Then we elaborate on the control structure of the model and at the end, we present the conceptual design of our agent.

Reasoning over Beliefs

The goal of the TPCT is to correctly classify and identify all contacts in the surroundings. To draw these kind of conclusions about contacts, the agent needs knowledge about their behavior. There are two ways to gather such information. The first is from the external world, e.g., the agent can watch the screen that displays information from sensors such as the radar system. Additionally, the agent can decide to perform actions that lead to more knowledge about the situation, such as activating radar or sending a helicopter to investigate a contact. The second method to gain information is through the internal process of reasoning about beliefs to deduce new beliefs. In the reasoning process, often multiple beliefs form the evidence for the formation of a new belief. Any uncertainty in the source beliefs will be transferred to the new belief.

The following rule is an example of how a new belief is derived using other beliefs and domain knowledge. The position of the contact that the agent is currently reasoning about is compared to the position of every other contact that is detected by the radar system. A new belief is created for every pair that indicates how certain the agent is that

they are within formation distance. Names of functions are depicted bold and names of parameters are not italic.

Determine_Within_Formation_Distance_Contact(X)
FOR ALL Y
  IF
  ($belief(position\_contact(X, P1), T1, S, C1)$,
   $belief(position\_contact(Y, P2), R1, S, C2)$,
   Position_Difference$(P1, P2, D)$,
   Certainty_Handling_Positions_Difference$(C1, C2, D, C3)$,
   Possible_Distances$(D, C3, [R])$,
   $M =$ maximum_distance_relevant_for_formation,
   $C4 = (number\_of\_[R \leq M])$ / $(number\_of\_[R])$)
  THEN
   Reason_Belief_Parameter$(within\_formation\_distance\_contact(X, Y),$
                $determine\_within\_formation\_distance\_contact, C4)$

The function **Position_Difference** calculates the distance between two positions, **Certainty_Handling_Difference_Between_Positions** calculates the certainty of the distance given the certainties of the positions, **Possible_Distances** returns all possible distances given the calculated distance and certainty, and the rule **Reason_Belief_Parameter** adds the time stamp and stores the belief in long-term memory. In this example, the latest beliefs about the positions are used. In other rules, beliefs are used that ever had a specific value, or those beliefs the agent is most certain about.

Control of Reasoning

Control is an important aspect of a cognitive agent; it determines when the agent does what. The TPCT has one main goal, which is considered a desire in the BDI model: to identify all contacts correctly. The three subtasks that the agent can perform in order to fulfill this desire are 1) processing information about contacts from the screen, 2) changing the activity of the radar system, and 3) sending the helicopter on observation missions to gain more information about a specific contact. These subtasks are the intentions of the BDI model that the agent can commit to.

A cognitive valid manner to determine when which intention becomes a commitment is to have events in the world trigger an intention. For example, when a contact suddenly changes its behavior, the attention of the agent should be drawn to this contact, regardless of the current intention. However, this type of control requires a parallel processing of all events in the world and a parallel checking of relevancy for all subtasks, which is very difficult to implement. That is why currently, we chose to implement a simpler, linear control system. To simulate parallel processing we let the agent alternately commit to

one of the three intentions. Within the subtasks, the control is also kept simple, e.g., in the first subtask all contacts on the screen that are stored in a random list are monitored consecutively.

The following reasoning rule is an example part of the simple control structure. It determines when the agent starts committing to a new intention.

Determine_New_Intention(I)
FOR ALL Y
  IF
  ($I = monitor\_contacts$,
   $belief(number\_of\_contacts\_monitored(X), T1, S, C)$,
   $X =$ maximum_number_of_contacts_to_monitor)
  THEN
   Start_New_Intention_Selection(I)
  ELSE IF
  ($I = monitor\_contacts$,
   $belief(number\_of\_contacts\_monitored(X), T1, S, C)$,
   $X <$ maximum_number_of_contacts_to_monitor)
  THEN
  (**Select_Next_Contact_To_Monitor**(),
   Reason_Belief_Parameter$(number\_of\_contacts\_monitored(X+1),$
   $determine\_new\_intention, 1)$)

The input parameter I is the current intention, the rule **Start_New_Intention_Selection** determines which intention is selected next depending on beliefs about contacts, and the rule **Select_Next_Contact_To_Monitor** selects the next contact from the list.

Conceptual Agent Design

We have made a conceptual design of the agent capable of performing the TPCT using the DESIRE (DEsign and Specification of Interacting REasoning components((Brazier et al., 2002) method. DESIRE's view of an agent is that of a composed structure consisting of interacting components. This conceptual agent model in DESIRE completes the formal model of the TPCT with the control of the different subtasks. DESIRE enables us to model the flow of information within the agent and between the agent and the external world. The agent components of the conceptual model are displayed in Figure 3.5, representing different kinds of tasks at different levels.

At the top level, there are two components: Warfare Officer and External World. This makes it possible to model communication between the agent and the external world. At the second level, three components are selected from the Generic Agent Model (GAM) (Brazier et al., 2000): (1) Own Process Control, (2) Maintenance of World

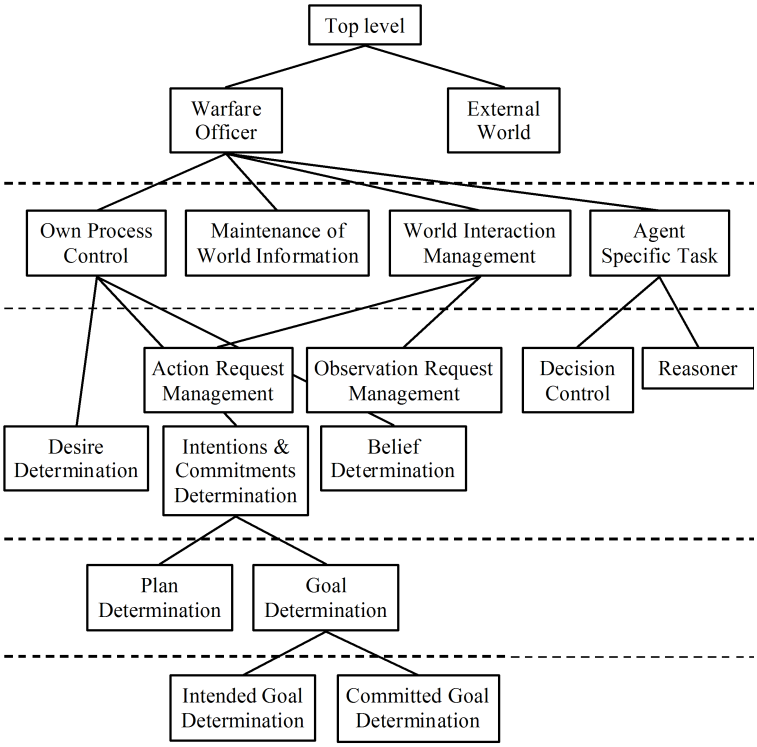


Figure 3.5: Process decomposition for the agent

Information and (3) World Interaction Management. The component Own Process Control, responsible for desire, intention and belief determination, is refined according to the addition to GAM for BDI models (Brazier et al., 2001). The component Maintenance of World Information stores the beliefs created by a subcomponent of Own Process Control. The last GAM component, World Interaction Management, manages communication with the external world via two subcomponents, Action and Observation Request Management. The fourth component at the second level, Agent Specific Task, performs tasks that require an internal decision to be made (Decision Control), and tasks that require reasoning about beliefs to derive new beliefs (Reasoner).

3.4.4 Translation Process

This section first introduces ACT-R and than continues with a description of the translation process of the reasoning rules and the control of these rules into the ACT-R architecture. Finally, it introduces the environment with which the resulting agent interacts.

ACT-R

ACT-R incorporates two types of memory modules: declarative memory and procedural memory. Declarative memory is the part of human memory that can store items; procedural memory is the long-term memory of skills and procedures. ACT-R consists of a central processing system, where the *production rules* representing procedural memory are stored and executed. The central processing system can communicate with several modules through buffers. One of those modules is the *declarative memory module* where memory items are stored. These memory items, called *chunks*, are of a specific chunk-type, which can be defined by the modeler. In a chunk-type definition, the modeler defines a number of slots that chunks of this type can assign values to. Chunks from the declarative memory module can be placed in the *retrieval buffer* if they match a retrieval request made by a production rule. A retrieval request must contain the requested chunk-type, and may contain one or more slot-value pairs that the chunk must match. The matching chunk is then placed in the retrieval buffer, so it can be read by a production rule. All buffers in ACT-R, including the retrieval buffer, can only store one chunk at a time, even when more chunks match the conditions of the request.

Reasoning Rules

In the following paragraphs, the implementation of the reasoning rules of the formal task model in the declarative and procedural memory modules is described.

Declarative Memory The cognitive model of the TPCT requires the ability to retrieve beliefs with specific time and certainty constraints. The rule **Determine_Within_Formation_Distance_Contact** for example, requires the latest beliefs about the position of two contacts. Because beliefs are not thrown away, there may be many older beliefs about the positions of these contacts. It is therefore necessary to retrieve the latest belief of both contacts in order to determine the current distance between them. In ACT-R, creation and retrieval times of chunks are registered, but these properties are not available to the modeler. There is also no feature that enables the agent to retrieve a chunk with the highest value of a slot.

The method ACT-R provides for retrieving chunks from memory is an activation function for chunks. The activation of a chunk indicates how easy it should be to retrieve it. Every chunk that matches a retrieval request receives an activation score based on three components: a *base level activation*, a *context component* and a *noise value*. The base level activation uses the number of representations and the time since the representations. Representations are the initial entry in the declarative module and a retrieval in the

declarative buffer. The context component is based on the activation of related chunks. The noise value adds a random factor to the activation value.

Our task model assumes that when the last belief is required for a reasoning process, that the last belief is always retrieved. In ACT-R, it is possible that an older chunk has been retrieved more often than the latest chunk, which results in a higher activation value. Although there are several parameters that can be set, the activation function cannot guarantee that the latest chunk is always the most active. To meet the requirements of the task model, it is therefore necessary to implement the retrieval process of those beliefs using LISP, the language ACT-R is built in.

Procedural Memory There are several theories about working memory (WM), most of which agree on the idea that multiple memory items can be stored at the same time in WM (Miller, 1956; Baddeley and Hitch, 1974; Hulme et al., 1995; Cowan, 2005). The number of items that can be stored in WM ranges from two to seven in these theories. ACT-R's theory of WM is that there can only be one chunk of memory stored in WM at a time. This representation of WM is not consistent with the most commonly accepted cognitive theories of WM, as well as with the formal task model based on the developed belief framework.

In the formal task model, several beliefs need to be in WM at the same time to be able to reason. For instance, different positions over time are compared in order to determine a contact's heading and speed. To implement such a rule in the ACT-R model, multiple production rules are needed to retrieve all required beliefs. Each production rule would have to draw a sub conclusion about the retrieved information so far. This method is inefficient for this task and it is inconsistent with the reasoning methods described by the naval experts. Therefore, we have tried two different methods to solve this problem. The first solution is to merge all the beliefs that need to be compared into one chunk. This solution however, still requires many production rules to retrieve all the necessary beliefs and undermines the ACT-R theory of WM being able to store only one chunk at a time.

The second method, used in the final model, consists of two parts: using the *goal buffer* for temporary storage and using LISP functions to retrieve beliefs. The goal buffer can hold just one chunk, so the different memory items need to be stored in the slots of the goal chunk. Many ACT-R modelers use this solution to be able to compare information items (see e.g., the tutorials of ACT-R 6.0). For example, in our task model several beliefs about the location of a contact at different time points form the antecedent of the logical rule that calculates the belief about the speed of a contact. These different locations can be stored in the goal buffer until all relevant locations are retrieved and the speed can be calculated. The second part of our solution is that we used LISP functions

instead of production rules to access chunks in the declarative module of ACT-R. LISP functions can be called within a production rule that is firing. Then, the LISP function can retrieve multiple beliefs at the same time, compare them, and return the result to the active production rule.

The many calculations the task model requires, e.g., for calculating the certainty of a new belief from the uncertainties of the beliefs it is derived from, is a second problem we encountered during translation. In most ACT-R models of low-level tasks, calculations are modeled in production rules. The addition of three and four for example, would take ten rules: one for the initialization, two for every addition step (increment by one, remember how much has been added) and one for the finalization. It would take many more production rules to calculate the speed of a contact from its positions over time. In addition, most of the calculations are not an exact representation of the cognitive processes of a warfare officer, but are a more abstract representation. It is therefore not necessary and not efficient to model all computations in production rules. We chose to use LISP functions for those calculations.

By executing the above processes in LISP functions instead of in production rules, the amount of production rules is reduced to half of the programming code of the agent. The other half consists of supporting LISP functions. The following ACT-R production rule is an example of how we implemented the task model rule **Determine_Within_Formation_Distance_Contact** and how we solved the problems described above.

```
(p determine-within-formation-dist-goal1-plan6
   =goal>
      ISA         commitment1
      goal        monitor-contacts
      state       determine-formation-distance
      contact     =contact
==>
   !bind! =within-dist (calc-within-formation-distance =contact)
   !bind! =next-step (if =within-dist *determine-formation*
                         *determine-classification*)
   =goal>
      state      =next-step
      result     =within-dist )
```

The antecedent of the rule requires that the agent is currently committed to commitment1; processing information about contacts from the screen, and that the agent is going to determine whether the current contact is within formation distance of any other contact. In the consequent, the agent uses a LISP function to find all contacts within formation distance (function `calc-within-formation-distance`). In this LISP function, multiple beliefs with the latest time stamp are retrieved, and calculations are performed to determine the distance. If a contact exists that is close enough to the current contact, the

agent starts with determining whether they actually are moving in formation. Otherwise, the next step is to determine the classification of the current contact.

Control of Reasoning Rules

As explained above we implemented a simple, linear control structure, in which the agent alternately commits to its various subtasks. We showed the reasoning rule **Determine_New_Intention**, which determines that after reasoning about a number of contacts one of the other subtasks becomes the agent's intention. We did not run into any problems when implementing these task model control rules in ACT-R. In ACT-R, the control structure is also linear: only one production rule can fire at a time. The antecedent determines which rule matches the current contents of the buffers. The current intention and current step in the plan can be stored in the goal buffer. The consequence of a fired rule can influence which production rule will fire next. In our agent, this was generally done by changing the contents of the goal buffer. The following ACT-R production rule shows how we implemented the rule **Determine_New_Intention**.

```
(p select-next-contact-goal1
   =goal>
      ISA            commitment1
      goal           monitor-contacts
      state          next-contact
      contact        =contact1
==>
   !bind! =contact2 (determine-next-contact)
   !bind! =new-intention (= (mod *contact-counter*
                                *max-number-of-contacts*) 0)
   =goal>
      plan           read-basal-info
      state          start-step
      contact        =contact2
      new-intention  =new-intention )
```

The antecedent of the rule requires that the agent is currently committed to monitoring contacts, and that the agent is going to select a new contact. In the consequent, the agent determines whether to select the next contact or to start a new intention. The `new-intention` of the goal chunk can be true or false, influencing the next production rule that can fire.

Coupling Environment

ACT-R has different modules for simulating visual and aural stimuli and vocal and manual actions. The visual module can be used to show letters and numbers in a window,

for example to simulate the Stroop-task. However, the current task requires the visualization of a naval scenario and the features of the visual module are too limited for this. Therefore, a coupling has been made between ACT-R and an external simulation environment: Overmars (2009). The simulation environment we developed in Game Maker consists of a screen that shows radar contacts. An additional screen displays detailed information about a contact when it is selected by a mouse click. This same window can be used to change the classification and identification value of the selected contact.

ACT-R and Game Maker communicate through two text files. In the first file, the ACT-R agent writes information and action requests, which Game Maker reads and performs. This can be, for instance, a request for detailed information, simulating a mouse click on a contact. In the other file, Game Maker writes the requested information and feedback about the performed action, which ACT-R reads and processes. The response of Game Maker to the mouse click would be to write the detailed information of the contact in the text file.

3.4.5 Results and Discussion

Now that we have implemented the TPCT model in ACT-R and coupled the resulting agent with a simulation environment, we can test the behavior of the agent. In this section, we will describe and discuss the results of the translation process and the coupling with Game Maker.

Before translating the task model into ACT-R, we developed a conceptual agent model in DESIRE that gave us a structured overview of the control and information flows within the task model. We find that the structure of this conceptual agent model matches well with ACT-R. Each of the components of the conceptual model (see Figure 1) can be identified in the ACT-R code. This made the conceptual model a useful structure to base the ACT-R model on.

In addition, the serial control was easily translated from the DESIRE model to ACT-R production rules. In ACT-R, one production rule can fire at a time. This principle is the same as we chose for our agent. However, if we had chosen a more complex control, we probably would have had more trouble translating the conceptual model to ACT-R. For example, a function that determines which contact on the screen deserves attention requires calculations. Since we have already shown that calculations are difficult to model in ACT-R, a more complex control system will be more difficult to implement.

During the implementation process, we encountered some problems when translating the reasoning rules of our model to ACT-R production rules. The ACT-R theory entails that one chunk can be stored in WM, that the chunk with the highest activation value is retrieved, and that it is difficult to combine low-level calculations and high-level

reasoning rules in one model. Our solution has been to implement those processes in LISP functions. However, by using so much LISP code and by not using the declarative memory module properly, the structures built into ACT-R that constitutes its theory of cognition are denied. It is likely that both theories are incomplete. ACT-R should support the storage of multiple chunks in the retrieval buffer at the same time since most researchers agree that human WM can store multiple beliefs. We should reconsider the many calculations performed in the rules of the cognitive task model.

A more practical issue we encountered during implementation concerns the number of chunks in ACT-R's memory. Every couple of minutes the agent is active, its number of chunks doubles. As a result, ACT-R becomes very slow when the agent tries to retrieve a chunk from memory. We partly solved this by creating fewer chunks during reasoning. In the original task model, every reasoning step resulted in a new belief. We identified types of beliefs that were never retrieved by the agent, and stopped adding them to ACT-R's memory. For example, only the result of a calculation is remembered, instead of all steps leading to this result. Furthermore, we deleted irrelevant chunks from the agent's memory. For instance, if the speed of a contact has been constant for a long time, the first and last beliefs about that speed provide all the information ever needed.

The coupling of the ACT-R agent and the simulation environment Game Maker consists of communication through text files; one file in which ACT-R writes its requests and. one file in which Game Maker writes its results. It is computationally impossible for both Game Maker and ACT-R to check the text file constantly for new information. The less often the text file is checked for new information, the slower the communication process becomes. However, if Game Maker and the agent would check it more often, the entire simulation would slow down.

We tried to find a balance by having Game Maker read the text file every second. The agent only reads the text file when it expects new information. Thus, when the agent communicates a request to Game Maker, it keeps reading the text file until the new information has arrived. In practice, this means that ACT-R reads the text file approximately every half second. In the future, we would prefer a different type of coupling that enables streaming of information, so the two parties do not have to actively check for new information.

3.4.6 Conclusion and Future Research

We implemented a cognitive task model in ACT-R and tested how it functioned. During implementation, we found that the DESIRE control model fits well with ACT-R, but that the model's reasoning rules that are based on the developed belief framework do not fit.

ACT-R is a cognitive architecture that consists of several modules. Two of those

modules are used for the agent: the declarative memory module and the procedural memory module. The communication with the chunks in the declarative memory module is not done by the use of the retrieval buffer, but through LISP functions. LISP functions are used to solve three issues. First, the task model requires the comparison of more than one belief, and ACT-R can only hold one belief in the representation of WM. Second, the task model requires the latest or most certain belief to be retrieved, and ACT-R does not offer a function to request beliefs with these specifics. Third, the task model describes many calculations, which are difficult to implement using ACT-R's production rules. Because only part of the theory of ACT-R matches the formal cognitive task model, this cognitive architecture is not very well suited for this kind of task model and this forced us to implement a great part of the agent in LISP code.

Overall, the implemented agent is slow. ACT-R is a software package that uses about half of the computer's CPU power, and Game Maker uses the other half. In ACT-R the slow speed is mainly due to the extensive search for specific beliefs caused by the exponential grow of chunks. To improve the performance of the agent it is necessary to extend the task model with a cognitive model for the decay of beliefs, or with a system that throws away beliefs that are not relevant for the task anymore.

In the future, we want to translate the cognitive task model to SOAR to research how well it matches with that cognitive architecture. This activity will yield a SOAR agent whose performance we can compare with the current ACT-R agent. To increase the cognitive validity of the task model we also want to focus on the development of a more cognitive plausible control system.

Acknowledgments

The authors like to thank Tibor Bosse and Karel van der Bosch for their guidance during this research project. Furthermore we like to thank Jan Treur for commenting an earlier draft, Willem van Doesburg for developing an initial version of the training environment, and the naval instructors of the Opschool for providing domain knowledge and for taking part in the experiment.

Research Paper

3.5 Implementing a Cognitive Model in ACT-R and Soar: A Comparison

Abstract

This paper presents an implementation of a cognitive model of a complex real-world task in the cognitive architecture Soar. During the implementation process there were lessons learned on various aspects, such as the retrieval of working memory elements with relative values, alternative approaches to reasoning, and reasoning control. Additionally, the implementation is compared to an earlier implementation of the model in the ACT-R architecture and both implementations are discussed in terms of cognitive theories.

This section is published as:
Muller, T. J., Heuvelink, A., and Both, F. Implementing a Cognitive Model in ACT-R and Soar: A Comparison. In Jung, Michel, Ricci and Petta (Eds.), *Proceedings of the 6th International Workshop on From Agent Theory to Agent Implementation* (AT2AI-6 2008) in conjunction with AAMAS 2008. May 13 2008, Estoril - Portugal.

3.5.1 Introduction

People performing tasks in uncertain and dynamic environments require much training in order to gain the necessary expertise. However, the nature of these tasks makes it hard to set up real-world training. An appropriate alternative for training decision-making in complex environments is scenario-based simulation training (Oser, 1999). To create a useful training, a simulation needs to represent the aspects of the real world that are vital for achieving the learning objectives. One of these aspects is human interaction; therefore, simulated entities that respond naturally and validly are needed. These entities, known as *agents*, can be used to simulate team members, opponents or bystanders. There is growing conviction and evidence that *cognitive agents* can be developed by capturing human cognitive processes in a cognitive model and implementing it in a cognitive architecture (Pew and Mavor, 1998; Ritter et al., 2003; Gluck and Pew, 2005).

An architecture poses constraints on the implementation of a model and therefore influences design choices. This paper reports the experiences of implementing the same formal cognitive model in two different cognitive architectures. First, the implementation of the model in the cognitive architecture Soar (Laird et al., 1987) is presented. This agent performs a real-world task in a complex environment. Implementing the cognitive model provides insights into the use of Soar for agent applications and it may be used to validate the model's behavior in future research. Next, the Soar implementation is compared to an earlier implementation of the same model in the cognitive architecture ACT-R (Anderson and Lebiere, 1998). This allows for the second goal of this paper: the comparison of Soar and ACT-R.

The next section presents the cognitive task and the formal model. Section 3.5.3 presents the ACT-R architecture and the implementation, BOA. Section 3.5.4 elaborates on the implementation in Soar, which resulted in the agent named Boar. The paper concludes with a comparison of both implementations on various aspects and their connection to the cognitive theories.

3.5.2 Cognitive Task and Model

The real-world task that has been modeled is the *tactical picture compilation task* (TPCT) from the naval warfare domain. In this task, a navy operator sees a large number of radar *contacts* on his display. Each contact indicates a detected vessel in the vicinity of the own ship. The identities and classifications of these vessels are unknown. The operator can obtain information on these tracks by monitoring the radar screen, such as speed, course, distance to own ship and adherence to shipping lanes. The task of the operator is to use this information to determine both the identity (e.g. hostile, friendly) and the

classification (e.g. frigate, fishing boat) of each contact.

A complete cognitive model of the TPCT is constructed using the principles described in this section. The model is based on an extended Belief, Desire and Intention (BDI) framework (Georgeff and Lansky, 1987). BDI facilitates the translation of domain knowledge into a model since domain experts tend to talk about their knowledge in terms similar to beliefs, desires and intentions. The domain knowledge needed to model the TPCT has been elicited from naval experts (Dam and Arciszewski, 2002).

In order to develop cognitive agents for training purposes, cognitive behavior that can vary in level of rationality needs to be modeled: agents that can perform a task on different levels of expertise are needed. To make this possible, a belief framework was developed (Heuvelink, 2007). Three arguments are added to beliefs: a time stamp, the source of the belief and a certainty level. BDI models usually throw away beliefs as soon as a new belief is created that causes an inconsistency. However, to enable reasoning over time, every belief is kept and labeled with the time of creation. The source and certainty labels make it possible to reason about information from multiple sources and with uncertainty, and reasoning can be done in both a rational and a biased way.

A belief $belief(P(A,V),T,S,C)$ has a predicate P with attribute A and value V, a time stamp T, source S and certainty C. Below is an example belief – there was an $identification$ of $contact1$ as $friendly$ with certainty 0.7, done by reasoning rule $determine\_id$ on time stamp 12:

$belief(identification(contact1, friendly), 12, determine\_id, 0.7)$

A cognitive model typically consists of declarative knowledge, denoting facts, as well as procedural knowledge, denoting reasoning rules. Besides modeling how to reason, it is also necessary to model the control on when to reason about what. The next subsection presents the format of the declarative and procedural knowledge and subsection 3.5.2 explains the control structure of the model.

Reasoning over Beliefs

The goal in the tactical picture compilation task is to correctly classify and identify all contacts. In order to fulfill this goal, the agent needs information about the contacts' behavior. There are two ways to gather such information. The first is from the external world, e.g., the agent can watch the screen that displays information from sensors such as the radar system. Additionally, the agent can decide to perform actions that lead to more knowledge about the situation, such as activating its radar or sending a helicopter to investigate a contact. The second method to gain information is through the internal process of reasoning about beliefs to deduce new beliefs. In the reasoning process, often multiple beliefs form the evidence for the formation of a new belief. Any uncertainty in

the source beliefs will be transferred to the new belief.

An example of this type of deduction is reasoning about formations. If a number of vessels have the same course and are close to one another (source beliefs), they might move in formation (new belief). Moving in formation is an indication that these vessels are frigates. Figure 3.6 contains an example rule that is part of reasoning about formations. The position of the contact that the agent is currently reasoning about is compared to the position of every other contact that is detected by the radar system. A new belief is created for every pair that indicates how certain the agent is that they are within a distance that can indicate a formation.

The function **Position_Difference** calculates the distance between two positions, **Certainty_Handling_Positions_Difference** calculates the certainty of the distance given the position certainties, **Possible_Distances** returns all possible distances given the calculated distance and certainty, and **Reason_Belief_Parameter** adds the time stamp and stores the belief in long-term memory. In this example, the latest beliefs about the positions are used. In other rules, beliefs are used that ever had a specific value, or those beliefs the agent is most certain about.

Determine_Within_Formation_Distance_Contact(X)
FOR ALL Y
  IF
  ($belief(position\_contact(X, P1), T1, S, C1)$,
   $belief(position\_contact(Y, P2), R1, S, C2)$,
   Position_Difference$(P1, P2, D)$,
   Certainty_Handling_Positions_Difference$(C1, C2, D, C3)$,
   Possible_Distances$(D, C3, [R])$,
   $M =$ maximum_distance_relevant_for_formation,
   $C4 = (number\_of\_[R \leq M])$ / $(number\_of\_[R])$)
  THEN
   Reason_Belief_Parameter$(within\_formation\_distance\_contact(X, Y),$
   $determine\_within\_formation\_distance\_contact, C4)$

Figure 3.6: Rule for determining if two contacts are within formation distance

Control of Reasoning

Control is an important aspect of a cognitive agent; it determines when the agent does what. In the TPCT there is one main goal, which is considered the navy operator's desire in the BDI model: to identify all contacts correctly. The three subtasks that the agent can perform in order to fulfill this desire are:

1. processing information about contacts on the screen;

2. changing the activity of the radar system; and

3. sending the helicopter on observation missions to gain more information about a specific contact.

The subtasks above are the intentions of the BDI model that the agent can commit to. A valid manner in cognition to determine when which intention becomes a commitment is to have events in the world trigger an intention. For example, when a contact suddenly changes its behavior, the attention of the agent should be drawn to this contact, regardless of the current intention. However, this type of control requires a parallel processing of all events in the world and a parallel checking of relevancy for all subtasks, which is hard to implement. This is why currently a simpler, linear control system is modeled. The agent alternately commits to one of the three intentions to simulate parallel processing. Within the subtasks, the control is also kept simple, e.g. in the first subtask all contacts on the screen are monitored consecutively.

The rule in Figure 3.7 illustrates a part of the simple control structure. It determines when the agent starts committing to a new intention. The input parameter I is the current intention, the rule **Start_New_Intention_Selection** determines which intention is selected next depending on beliefs about contacts, and the rule **Select_Next_Contact_To_Monitor** selects the next contact from the list.

Determine_New_Intention(I)
FOR ALL Y
  IF
  ($I = monitor\_contacts$,
   $belief(number\_of\_contacts\_monitored(X), T1, S, C)$,
   $X =$ maximum_number_of_contacts_to_monitor)
  THEN
   Start_New_Intention_Selection(I)
  ELSE IF
  ($I = monitor\_contacts$,
   $belief(number\_of\_contacts\_monitored(X), T1, S, C)$,
   $X <$ maximum_number_of_contacts_to_monitor)
  THEN
  (**Select_Next_Contact_To_Monitor**(),
   Reason_Belief_Parameter$(number\_of\_contacts\_monitored(X+1),$
   $determine\_new\_intention, 1)$)

Figure 3.7: Rule for determining a new intention

3.5.3 BOA

In order to execute the model that was presented in the previous section, a cognitive agent needs to be implemented; cognitive architectures are a suitable platform for this purpose. Such an architecture specifies a fixed set of processes, memories and control structures (Lewis, 2001) that define the underlying theory about human cognition. The architecture limits implemented cognitive models by this set and consequently imposes its cognitive theory on these models: it should make correct models easier and incorrect models harder to build. Moreover, the actual behavior of the agent is influenced by the architecture (Jones et al., 2007).

The presented model has already been implemented in the cognitive architecture ACT-R (Both and Heuvelink, 2007) – this implementation was named BOA. Since this research has been done earlier, several new developments in ACT-R are not taken into account (Anderson, 2007). However, the insights reported here are nevertheless of interest from an agent-application perspective: several of the issues mentioned in this paper have been changed in the latest version of ACT-R. These developments in ACT-R seem to support our experiences that the architecture was too restrictive on some aspects.

ACT-R

The theory of ACT-R incorporates two types of memory modules: declarative memory and procedural memory. Declarative memory is the part of human memory that can store items; procedural memory is the long-term memory of skills and procedures. ACT-R consists of a central processing system, where *production rules*, representing procedural memory, are stored and executed. The central processing system can communicate with several modules through buffers. One of those modules is the *declarative memory module* where memory items are stored. These memory items, called *chunks*, are of a specific chunk-type, which can be defined by the modeler. In a chunk-type definition, the modeler defines a number of *slot*s that chunks of this type can assign values to. Chunks from the declarative memory module can be placed in the *retrieval buffer* if they match a retrieval request made by a production rule. A retrieval request must contain the requested chunk-type, and may contain one or more slot-value pairs that the chunk must match. The matching chunk is then placed in the retrieval buffer, so it can be read by a production rule. All buffers in ACT-R, including the retrieval buffer, can only store one chunk at a time, even when more chunks match the conditions of the request. If more chunks are available, an activation function defining the accessibility of chunks is used to select a single candidate.

Implementation Issues

The implementation of the cognitive model in ACT-R resulted in three main observations. The first focuses on the limit of one chunk in the retrieval buffer. The model prescribes access to multiple beliefs in the working memory at the same time in order to reason over them. For example, different positions in time are compared in order to determine a contact's speed. The ACT-R implementation supported this by using the goal buffer for temporary storage and LISP functions to retrieve beliefs.

The second observation was the fact that retrieving a belief with specific features (for example, the belief created last, i.e. with the highest value for the *time* slot) is not guaranteed by using ACT-R's activation function. For example, the agent often uses the latest position of a contact, so he needs the latest belief with predicate `position-contact` for a specific contact. It may however be that an older chunk has been retrieved more often than the latest chunk, resulting in a higher activation score and subsequently the older chunk being retrieved. As a solution, LISP functions were created as substitute to the activation function.

The third issue is about the many calculations the cognitive model requires: these can only be modeled in a low-level manner, making it inefficient to implement them in the architecture. For example, calculating the speed of a contact from its positions over time would require many production rules, while it would not represent the actual cognitive processes of a warfare officer. Here too LISP functions were used for these type of calculations. As a result of this problem and the previous problem, about half of the programming code consists of ACT-R production rules and the other half of supporting LISP functions.

Control

Control in the context of BDI agents aims at specifying the commitment of the agent at a certain time. The intentions to which the agent can commit and the type of control in the cognitive model were described in section 3.5.2. The BOA agent implements a simple, linear control system. The agent commits alternately to each intention and within the intention of processing screen information, the contacts are monitored sequentially. This is illustrated by the rule in Figure 3.8.

The rule requires the agent to be committed (`commitment1`) to monitoring contacts and be ready to select a new contact to monitor. This new contact is determined by the user-defined LISP function `determine-next-contact`, which loops through the list of contacts. The `*rate-other*` variable defines the number of contacts after which the

```
(p select-next-contact-goal1
   =goal>
      ISA         commitment1
      goal        monitor-contacts
      state       next-contact
      contact     =contact1
==>
   !bind! =contact2 (determine-next-contact)
   !eval! (determine-rate-other-desires)
   !bind! =eop (= (mod *counter* *rate-other*) 0)
   =goal>
      plan        read-basal-info
      state       start-step
      contact     =contact2
      eop-marker  =eop
)
```

Figure 3.8: ACT-R code for intention selection

agent switches to another intention: if this number is reached, the end-of-process marker (`eop`) is set to true. The agent will then consider committing to sending the helicopter, followed by considering to commit to changing the radar. After these considerations and, possibly, reasoning and actions, the agent continues monitoring contacts. Reacting to events in the environment is limited to altering the order of the list of contacts in the 'monitor contacts' intention: if a contact has been identified by the helicopter, that contact is moved to the top of the list to force the agent to monitor it next.

3.5.4 Boar

This section presents the Boar agent, which is the implementation of the model from section 3.5.2 in Soar. The next subsection will explain this architecture in more detail and subsection 3.5.4 describes several implementation issues.

Soar

Soar, like ACT-R, is a well-known cognitive architecture. Soar defines the world as a large problem space with states and goals. It considers behavior as movement in the problem state by performing actions, either internal (mental activity) or external (observable in the environment). In Soar, this is done by *operators*; in a single cycle, more operators can be proposed, one of these is selected and eventually applied, changing the state of the environment. Goal-directed behavior states that the agent will choose those operators that lead to a goal state (Lehman et al., 2006).

The memory structure of Soar is somewhat similar to that of ACT-R. It specifies two types of memory: the long-term memory, consisting of procedural, semantic and episodic knowledge, and the *working memory*, corresponding to ACT-R's declarative memory module. The working memory consists entirely of working memory elements (WMEs), which are attribute-value pairs. The attributes of a WME need not be defined beforehand, as is the case with the slots of a chunk. Additionally, the number of WMEs that can be accessed at one moment is not limited – there is no such thing as a retrieval buffer in Soar.

In long-term memory, the procedural knowledge is primarily responsible for the behavior of an implemented model and is defined in terms of *productions*. When conditions apply, a production either proposes the execution of an operator or it executes some reasoning independent from an operator – both may result in changes to the working memory. The difference lies within the persistence of the changes: a WME that was created by an operator will stay in working memory until an explicit change is made. A production without operator reference, also called *elaboration*, creates WMEs that only exist as long as the conditional part of the elaboration matches. The first is said to have operator support or *o-support*, while the latter has instantiation support or *i-support*.

Soar's productions fire in parallel: all productions that have one or more matches for their conditional part in the current state will execute. Consequently, many operators may be proposed at a single moment. Which operator is selected is resolved by means of preferences, which can be added to an operator.

The fact that Soar allows more production rules to fire simultaneously is in contrast to ACT-R's procedure: here, only one production rule can fire at a single moment. If more chunks are available for retrieval by this production, the activation function determines beforehand which chunk is picked.

If a task is too complex to solve by simply adding some beliefs to the working memory, it can be decomposed in subtasks. An example is reasoning about the usage of the helicopter. In order to decide which contact the helicopter is sent to, all contacts are scored. The rule for proposing the operator that scores a single contact is shown in Figure 3.9. If this operator is chosen, there is no immediate score available to be added as belief: it needs to be calculated. As a result, there is an impasse and a new substate is created which has the goal to calculate this score. Various operators are available to calculate a part of the score; after each operator calculated its part, the score is available and the attribute `heli-score` will have a value. Consequently, the operator shown in Figure 3.9 will be retracted, having achieved its goal.

```
# Propose to score a contact
sp {consider-heli*propose*score-contact
   (state <s> ^name consider-heli
              ^contacts.contact <contact>
              ^top-state.constants <const>)
   (<constants ^max-distance <max>)
              # No score, no visual id
      # and in range of self
   (<contact> -^heli-score
              -^visual-id-belief
               ^distance-to-self <= <max>)
-->
   (<s> ^operator <op> + =)
   (<op> ^name score-contact
         ^current-contact <contact>)
}
```

Figure 3.9: Soar code for proposing to score a contact

Implementation

Retrieving Beliefs In section 3.5.2 we argued that the model needs to be able to reason over time. For example, in Figure 3.10 the latest position of the own ship (`self`) is retrieved, i.e. the belief with the maximum value for the `time` attribute. This is an example where a belief with a *relative* value is needed for a certain attribute and the absolute value is of no importance. However, it is not easy to match such a belief if no absolute value is available. We tackled this problem by ordering the beliefs with greater-than and smaller-than relations. In order to retrieve the last-but-one belief a new production needs to be added, another for the last-but-two, and so on.

Reasoning Efficiency Two alternatives arise when generating new knowledge by means of reasoning (i.e. internal action). As explained in subsection 3.5.4, there are two ways of adding new elements to the working memory: either with o-support or with i-support. The advantages of using i-supported WMEs are:

1. they are created automatically if the conditions or the creating production apply in the current state;
2. they are removed if this not the case anymore and thus are not valid in the current state; and
3. they are updated automatically if new information is available.

The advantage of o-supported WMEs is that they are only created when the operators are proposed explicitly, so only at these times some reasoning is done.

```
# Calculate distance of contact to self.
sp {boar*elaborate*contacts*distance-to-self
   (state <s> ^name boar
      ^beliefs <beliefs>
      ^contacts.contact <contact>)
   # Retrieve latest position of self
   (<beliefs> ^belief (^predicate position-contact
                       ^attribute self
                       ^time <time>
                       ^value <self-pos>))
  -(<beliefs> ^belief (^predicate position-contact
                       ^attribute self
                       ^time > <time>))
   # Retrieve position of contact
   (<contact> ^position-belief.value <pos>)
-->
   (<contact> ^distance-to-self (float (exec
                      calcPositionDifference
                      <self-pos> |;| <pos>)))
}
```

Figure 3.10: Soar code for retrieving the distance between contact and own ship

If the beliefs that are used for reasoning stay the same for some time, it is more efficient to use elaborations and thus create i-supported WMEs, because if operators are used for this reasoning, they may perform the same reasoning steps multiple times. If beliefs change continuously, the use of elaborations may become computationally expensive, because they perform the reasoning at every change, even if the results are not used. In this case using operators and o-supported WMEs is more efficient. There is no clear procedure for choosing i-support or o-support: one should think about the trade-offs for every situation in order to pick the most efficient option.

To draw the differences between creating i-supported and o-supported WMEs more clearly, an implemented example of both types is given. First consider the production in Figure 3.10. It continuously creates an i-supported WME with the distance of contact `<contact>` to the own ship. Every time a new position is observed, either from the own ship (`<self-pos>`) or the contact (`<pos>`), the conditional part of the production for the old WME does not match the current state anymore and the WME is discarded. At the same time, the newly observed information is used to create a new WME for the contact with the `distance-to-self` attribute, and thus the knowledge has been automatically updated. For this reasoning step the i-supported option has been chosen, since the distance is needed continuously for other reasoning. Using an operator would mean that this operator needs to calculate this distance every time the information is

needed.

Now consider the production in Figure 3.9. This production sets in the creation of knowledge with o-support: it proposes an operator that, when applied, will assign a certain score to a contact. This score is used for considering to send a helicopter to the contact for identification. This scoring is only done incidentally, which makes the use of an operator a better choice. An elaboration would update this score continuously, even while it is not needed most of the time.

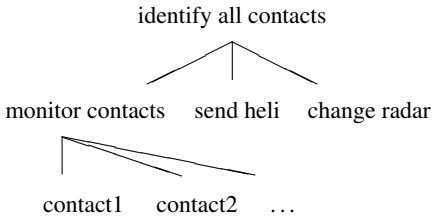


Figure 3.11: Overview of intentions

Control The Soar architecture does not provide the means to easily keep a list of contacts, which makes it hard to implement a sequential control for committing to the three intentions described in subsection 3.5.2. Alternatively, it easily allows the creation of subgoals, as explained in subsection 3.5.4. By defining the monitoring of every contact as a subgoal of the 'monitoring contacts' intention, the structure of goals and subgoals becomes as shown in Figure 3.11.

The commitment to one of the three intentions is decided as follows: if an event triggers the intention to send a helicopter or change radar activity, the agent commits to this intention. Otherwise it will first monitor all contacts, then consider sending the helicopter and finally consider changing radar activity. The linearity of this cycle is forced by explicitly remembering the control status in WMEs. Figure 3.12 shows the commitment to monitoring contacts: when a new cycle starts, the time is saved in the `start-process-passive` attribute. Then every contact not checked after this time (`-^checked > <starttime>`) is monitored and gets tagged with a new time, until all contacts have been checked this way. After completing the helicopter and radar intentions, a new cycle is started. The order of checking contacts is random and may be different each cycle.

An example of an event triggering the 'send heli' intention is shown in Figure 3.13. It simply states that if the helicopter is airborne and does not have a mission, for example

```
sp {boar*propose*process-passive-information
   (state <s> ^name boar
              ^start-process-passive <starttime>
              ^contacts.contact <contact>)
   (<contact> -^checked > <starttime>
              ^id <contact-id>)
-->
   (<s> ^operator <op> + =)
   (<op> ^name process-passive-information
         ^current-contact <contact>)
}
```

Figure 3.12: Soar code for proposing to monitor a contact

just after identifying a specific contact, the agent should commit to reasoning about what it should do. This commitment can break into the aforementioned cycle at any time the conditions apply.

```
# A heli is considered when it has no mission and is airborne.
sp {boar*propose*consider-heli*airborne
   (state <s> ^name boar
              ^helis.heli <heli>)
   (<heli> ^id <heli-id>
           # Heli has no mission and is airborne
           ^mission free
           ^status airborne)
-->
   (<s> ^operator <op> + =)
   (<op> ^name consider-heli
         ^heli <heli>)
}
```

Figure 3.13: Soar code for the event-driven selection of the 'send heli' intention

External Functions The simulation environment for performing the tactical picture compilation task is created in Game Maker (Overmars, 2009). It simulates a radar screen with information about the contacts in the surroundings. The simulation environment reacts on certain actions, e.g. clicking on a contact will provide detailed information about it.

For letting Soar communicate with this Game Maker environment an interface is needed, which is implemented in Java. The actions of the agent are written to a text file by the Java interface, read by Game Maker and consequently performed in the environment.

An example of an agent action is the request for detailed information, which simulates a mouse click by a human. Any input from the environment, such as a new contact or the position of the own ship, is written to a text file by Game Maker, read by the Java interface and presented as input for the agent. This form of communication slows the execution of the agent down, since it continuously waits for reactions from the environment.

To perform complex calculations, user-defined functions in Java are needed, similar to one of the issues when implementing the formal model in ACT-R (see subsection 3.5.3). These functions are called from inside the agent, but can only be used in the actions of a production. Consequently, if a calculation needs to be performed as part of the condition of a production, it has to be executed by another production and the result needs to be made available through the working memory.

3.5.5 Conclusion and Discussion

This paper presents an agent built in the cognitive architecture Soar, capable of performing a complex real-world task. The implementation is based on a formal model of the task and has previously been implemented in ACT-R. The remainder of this paper will present the lessons learned on several aspects of agent practice and links them to cognitive theories.

Two implementations of a single cognitive model give only one point of view: a different model may have different demands, especially when a different framework is used. Additionally, the implementations have not been validated – further work in this direction may consist of experiments with subject-matter experts comparing the performance of BOA, Boar and humans performing the tactical picture compilation task.

Nevertheless, this paper shows that one should consider the functionalities requested by the model and the possibilities an architecture offers to implement those demands.

Working Memory Access Most of the cognitive theories about human working memory agree on a storage capacity of multiple but limited amount of items (Miller, 1956; Baddeley and Hitch, 1974; Hulme et al., 1995; Cowan, 2005). This assumption was used in designing the cognitive model: several rules in the cognitive model need multiple beliefs at the same time for reasoning (an example of such a rule is in Figure 3.6).

The ACT-R theory used for implementing BOA proved to be too restrictive: access to only one chunk at a time is allowed. In order to access more beliefs, a work-around was used in the ACT-R implementation. On the other hand, Soar does not limit the number of accessible working memory elements, so this did not pose any problems implementing Boar. Different approaches to the working memory theory result in different types of behavior: if only a limited number of elements is accessible, reasoning will be restricted

to these elements, which can cause a different way of acting than when all elements are available.

Retrieving Beliefs In order to reason over time the retrieval of specific beliefs with a relative value is needed, such as the 'last' or 'one-but-last' belief of some kind – a capability humans apply unconsciously. Unfortunately, this type of retrieval operator is not yet available in either architecture.

In ACT-R the working memory items are retrieved by means of an activation function. However, this function does not guarantee the retrieval of a memory item based on such a relative value. The solution was to create LISP functions for retrieving beliefs. Soar allows ordering the beliefs in the conditional statement of a production rule, making it possible to retrieve beliefs with a relative value. However, operators need to be created for each relative value, making the translation from model to Boar somewhat inefficient.

Control A linear control was modeled in favor of event-driven parallel control. This choice was made in order to simplify the process of committing to an intention, even though human decision-making will be more reactive to cues from the environment. ACT-R's sequential execution of production rules fits this simplified model, but as a result the BOA agent reacts slowly on important changes in the environment, because the agent's behavior cannot be interrupted by these external events (Heuvelink and Both, 2007). In Soar the sequential execution of plans is forced by letting production rules fire in an explicit order (as shown earlier in Figure 3.9), but this architecture more easily allows event-driven control.

Calculations The tactical picture compilation task contains many situations in which the human expert makes estimations, for example on how close a ship is to the the own ship or whether ships are moving in formation. Currently there is no method available to model the process of these estimations. Instead, the estimations are replaced by exact calculations and made into an 'estimation' by adding the notion of uncertainty to the resulting belief. Modeling these calculations as cognitive tasks in an architecture would require an infeasible amount of productions, without actually copying human behavior. Therefore, the execution of complex calculations is done externally by LISP functions or Java methods.

Speed Humans are able to use multiple beliefs that were gathered over time for reasoning. This is represented in the belief framework by adding a time tag to every belief and storing all beliefs in memory. As a result, the agent can access multiple beliefs over time

for reasoning. For example, it can access several beliefs about the position of a contact to reason about the contact's speed and movement behavior.

Unfortunately, this creates a practical problem: there is an exponentially growing amount of beliefs, which means no system will eventually be able to cope with the resulting CPU-expensive searches for the necessary beliefs during real-time simulation. It is necessary to deal with this more efficiently. Even though certain facts in the past need to be remembered by the agent, it is not necessary to remember every specific detail, which is the case in this implementation. Humans do not remember every detail exactly, but compress their memories by conceptualizing or clustering them. Future agents that incorporate the belief framework used in this research will need some form of compression or smart discarding of beliefs to copy this behavior. We are currently developing a method to cluster and abstract beliefs over time, sources and certainties, in order to form a more realistic model of episodic memory.

Clearly, this problem has its effect on the implementations. The ACT-R agent becomes slow over time, even though some functionality to remove unimportant beliefs had been implemented. This slowness makes the observed behavior of the agent not very human-like, especially in reacting to changes in the environment (Heuvelink and Both, 2007). On the other hand, Boar has been used in a demonstration of about twenty minutes, in which the agent showed no reduction in speed. To draw general conclusions about the performance of both architectures, further research is needed.

Acknowledgments

This research has been supported by the research program 'Cognitive Modeling' (V524), funded by the Netherlands Defense Organization.

Chapter 4

Memory Component

4.1 Introduction

In the previous chapter we introduced a belief component for modeling rational and biased behavior, for which it was proposed to attach a *time* stamp, a *source* label, and a *certainty* value to beliefs. The agents we implemented embedding this belief component received an unlimited belief base, and no devision was made between long term memory and working memory: all beliefs ever stored were always available to reason upon.

During task execution, beliefs were constantly added to the belief base. The consequence of this rapidly growing, single belief base was that querying the database for a specific belief required ample processing time. Because a large part of the agent's behavior involved reasoning upon stored beliefs, this increase in query time had as effect that the agents slowed down over time. This effect was substantial for the agent BOA, implemented in ACT-R (see Section 3.4). The ad-hoc solution implemented in BOA to deal with this increase in execution time was to throw away beliefs of which it was determined from the task perspective that they would not be required at any later time point. The agent Boar implemented in Soar (see Section 3.5) did not noticeably slow down over time. However, it is likely that when the period during which Boar has to operate increases, its execution time will eventually increase as well.

In this chapter we focus on solving this increase-in-execution-time problem of agents embedding our belief component by developing an accompanying memory component. Besides the goal of decreasing the time required to query the agent's belief base, we also want the memory model to be able to support varied behavior. In particular, we want to develop it in such a way that it can provide 'perfect' memory and rational retrieval, as well as 'human-like' memory, which includes forgetting and biased retrieval of beliefs.

In the next section we provide a short overview of ideas on the functioning and structure of human memory. Subsequently, we describe the ideas that we have selected to embed in our memory model. Once more we would like to point out that our research is a typical example of *design* science (Section 1.2.1). We do not aim to present a universal cognitive valid model of human memory. Merely, we want to model a practically usable construct, called memory, which enables an agent to deal with the encoding, storing, and retrieving of time, source, and certainty labeled beliefs in a variety of human-like ways.

4.1.1 Human Memory

Memory enables humans to store information, to retain it, and to retrieve it. Multiple memory classifications are possible.

A common way to classify various types of memory is on the duration of memory retention, by which usually three types are distinguished: sensory memory, short-term memory and long-term memory. *Sensory memory* denotes the ability to retain for a very short period of time ($<$ 1 sec), accurate, but unprocessed, impressions of sensory information. *Short-term memory* (STM) denotes the ability to retain for a somewhat longer, but still short period of time ($<$ 1 min), a small amount of conscious, readily available information. The opinions about the exact capacity of STM vary, but a frequently cited number of elements it can hold is 7 ± 2 (Miller, 1956). *Long-term memory* (LTM) denotes the ability to store and retain large amounts of information for very long periods of time ($<$ life time). Information stored in LTM is not readily available, but needs to be retrieved.

The information retained in Sensory memory resides in some kind of sensory buffer and is not conscious. Because we are interested in a memory model for conscious information in the form of beliefs, this memory type is not relevant. Therefore, we do not further discuss this type of memory.

The STM concept is generally treated as similar to working memory (WM), which is a more specific model of the temporary storage of readily accessible information. The WM model was proposed by Baddeley and Hitch (1974) and consisted originally of three components: the central executive, the phonological loop and the visuo-spatial sketchpad. Recently, Baddeley (2000) extended the WM model with an additional component: the multi-modal episodic buffer. The central executive is held responsible for coordinating the other three 'slave' systems.

The LTM concept is usually subdivided into *declarative* (explicit) and *procedural* (implicit) memory (Anderson, 1976). Declarative memory consists of information that is explicitly stored and retrieved, while procedural memory consists of information that cannot be explicitly verbalized, but is learned implicitly, e.g., how to ride a bike.

Declarative memory is usually further subdivided into *semantic* memory and *episodic* memory (Tulving, 1972). Semantic memory concerns information independent of a context, e.g., an oak is a tree, while episodic memory concerns information specific to a particular context such as a time and place, e.g., the tree that stood in our garden in my childhood was an oak. So, semantic memory is used to encode abstract knowledge about the world while episodic memory is used to encode 'personalized' knowledge. However, these two memory types are related: semantic knowledge is not innate, but somehow emerges from episodic knowledge stored over time.

4.1.2 Selecting an approach

We want to develop a memory model that decreases the query time of the agent's belief base, and that supports rational as well as biased behavior. In this section we first elaborate on the way we propose to model memory so it decreases the query time of beliefs. Next, we discuss the way in which we plan to embed human-like aspects in the memory model. Subsequently, we compare the proposed model to the memory models embedded in current integrated architectures, and last to human memory.

Decreasing Query Time

The reason why the query time of the belief base increased was because beliefs were only added. One of the motivations to not throw away beliefs was that for the modeling of rational behavior, it should be possible to retrieve when what was believed. However, even for rational task behavior it is not required to be able to retrieve all beliefs, but only those that are relevant given the task perspective. For biased behavior it is definitely not required to be able to retrieve all beliefs, since humans simply forget things.

Here we propose a memory model that embeds mechanisms to make the beliefs possibly required for the task readily available for retrieval, while others (e.g., redundant ones) become much harder to retrieve. For this, we propose to divide memory into long-term memory (LTM) and short-term memory (STM), also referred to as working memory (WM). In the previous chapter all beliefs ever formed always held at the current time t: $holds\_at(b, t)$. In this chapter all beliefs are stored in LTM as $holds\_at(b, t*)$, with t* being the time they were held in STM. From this it logically follows that STM is made up by the beliefs b that $holds\_at(b, t)$, where t is the current time.

In addition, we introduce the concept of *complex* beliefs, which can be formed by aggregations on other beliefs, and are therefore also referred to as *aggregated* beliefs. Complex beliefs are introduced because we realize (due to our previous studies) that often specific types of beliefs are required, e.g., the *last* belief about a certain predicate

and attribute, or the *most certain* belief. In the previous chapter, these types of specific beliefs had to be deduced at the moment they were required by querying the belief base. However, many of them stayed, at least for a while, the same. When we would store the result of a specific query, this would decrease the query time for specific types of beliefs. This is what we use complex beliefs for. Complex beliefs denote specific types of beliefs that cluster basic information at a level at which reasoning rules can directly operate on them. The formation of complex, or aggregated beliefs speeds up the query time. Not because fewer beliefs are present (the aggregated beliefs only add to the amount), but because they form a significantly smaller top level of beliefs that are actually required for task execution and which can be queried in a faster way.

Complex beliefs are formed by *complex aggregations* that function as unconscious reasoning templates for deducing specific complex beliefs from other beliefs. Complex aggregations are controlled by setting an *in-task-focus* predicate to an aggregated belief type. When a complex belief is in-task-focus, i.e., required for task execution, the memory model first attempts to retrieve it from its belief base. When this is not possible, the complex belief is deduced. For this, the complex aggregation has to query the belief base, which requires the same amount of time as before.

Various mechanisms are possible to speed up this process: we decided to speed it up by attaching an *availability* value to all the beliefs. This value is currently based on the base-level activation-formula embedded in ACT-R, which is claimed to be cognitive valid (see Anderson and Schooler, 1991). In short, this formula attaches a higher availability value to recent and frequently used beliefs. When specific types of beliefs are important for the task execution, they are frequently placed in-task-focus, and as such have a high availability value. The availability value of beliefs that are not important, and thus not reused, will decrease over time. As such, this process mimics in a neater way the BOA solution of throwing away the redundant beliefs.

Implementing Human-Like Aspects

The availability value introduced can be used to influence the likelihood of retrieving a belief by setting a specific retrieval threshold, and can thus be used to model forgetting of beliefs. In addition to this basic aspect of human memory, we would like to model more sophisticated aspects. In particular, the phenomenon that over time details of an event are forgotten and only abstractions are remembered, and that semantic memories are formed out of episodic memories. We propose to model these aspects by using the availability value attached to beliefs in combination with the notion of *abstract* beliefs.

Aggregated beliefs can be *complex* beliefs as introduced above, but also *abstract* beliefs. *Basic aggregations* form abstract beliefs by abstracting from specific values of

arguments. The basic aggregations are controlled by setting an *in-memory-focus* predicate to an abstraction type. When a specific abstraction type is in-memory-focus, that abstracted belief type is deduced for the beliefs active in WM.

Abstract beliefs receive a slightly higher base-level activation than the beliefs they are abstracted from. This fact and the fact that the abstraction of a variety of beliefs can result in the same abstract belief, causes abstract beliefs to be more available than more detailed beliefs. As such, beliefs of which details are 'forgotten', i.e., abstract beliefs, might still be retrievable while the basic beliefs are not. In addition, when abstract beliefs have abstracted from the time, source, and certainty labels of beliefs, they have effectively made a transformation from an episodic to a semantic memory.

The memory model can be used to model two types of biased behavior. First, the memory model is able to display biased retrieval, because ACT-R's base-level belief activation-function accounts for order effects. Second, the model is able to support biased behavior because it supports the retrieval of abstract beliefs. The memory model's in-task-focus predicates determine which beliefs are retrieved for task execution. These predicates are in turn determined by the conscious reasoning rules that are selected for execution. These reasoning rules can request as input 'perfect' queried information (basic or complex beliefs), which makes them rational and expensive. On the other hand they can also request as input information that can retrieved quick and dirty (abstract beliefs), making them cheaper to execute, but possibly leading to biased behavior.

Comparison with Integrated Architectures

Although all integrated architectures embed a memory, none embeds a memory model similar to the one proposed by us. The architecture that comes the closest is CLARION, which offers an episodic memory as part of its non-action-centered subsystem in which beliefs can be stored with a time stamp (and other labels) attached. However, this architecture constitutes a dual-layer model of cognition, while we focused on a single-layer model. In addition, CLARION has not been applied to model this high-level type of tasks, the most complex tasks it has modeled are the Tower of Hanoi and minefield navigation tasks.

Soar has been applied to model high-level tasks, and its latest versions embed an episodic memory. However, this memory does not encode memories that are temporally indexed, so does not support the reasoning over beliefs over time in an absolute way (Nuxoll and Laird, 2004). In addition, it does not account for forgetting and generalization (Nuxoll and Laird, 2007), although in related work the working memory elements of Soar were extended with an activation value, which could be used for modeling the first aspect (Nuxoll et al., 2004).

The fact that current architectures do not embed the proposed memory model does not mean that it is impossible to implement it in them. We have implemented the developed belief framework in Soar and ACT-R and it would probably be possible to implement this memory model in other architectures. However, this would require a lot of work and the circumvention of the memory processes that they do embed.

Memory Model compared to Human Memory

Human sensory memory retains sensory information which is not yet processed by the cognitive system. We do not take this type of memory into account, because we are interested in a memory for conscious information, i.e., beliefs. In addition, at this moment we do not incorporate procedural memory. We decided to focus on declarative memory first because the situational assessment task is mainly a conscious, knowledge-intensive task, and not a, e.g., motor task.

We do make a division between an agent's long term memory (LTM), and its short term memory (STM) (in Section 4.2) or working memory (WM) (in Section 4.3). We use these names as constructs to denote whether the beliefs are currently available to the agent to reason upon (present in STM/WM) or whether they were once available and now stored away, with the possibility to retrieve them (present in LTM). We do not aim at participating in a discussion on the nature of the various types of memory, and on the best way to model them. For a good overview of current approaches within cognitive science to modeling WM, see Miyake and Shah (1999).

Because the beliefs in our belief framework receive a time, source, and certainty label, they resemble episodic knowledge more than semantic knowledge. Actually, we propose a memory model for agents that is purely episodic in nature. However, semantic knowledge is accounted for by incorporating aggregations that combine and abstract from episodic memories. As such, our memory model not only supports the existence of semantic memories, it supports their formation as well.

4.1.3 Chapter Overview

This chapter embeds two papers. The first paper (Section 4.2) introduces the belief algebra that constitutes a generic approach to form any specific type of aggregated (complex or abstract) belief out of stored beliefs. The generic approach entails that for forming an aggregated belief, a specific constraint is taken into account that defines certain properties for (a subset of) the belief arguments.

In the second paper (Section 4.3) we use the belief algebra to develop the memory model for agents incorporating beliefs with times, sources, and certainties attached. This

model controls the complex aggregations that form complex beliefs by incorporating in-task-focus predicates; it controls the basic aggregations that form abstract beliefs by incorporating in-memory-focus predicates. In addition, it embeds a mechanism that determines and attaches an availability value to all the beliefs. This availability value is used to model biased retrieval of beliefs, as well as the forgetting of beliefs.

Additional Remarks

This chapter embeds beliefs that are slightly different notated than before. Previously, we wrote a belief as $belief(p, a, v, t, s, c)$ or $belief(P(A, V), T, S, C)$. Both denoted, using variables, that it is believed by the agent at time T, based on source S and with certainty C, that predicate P holds for attribute A with value V. In this chapter, beliefs are denoted by $belief(p, o, v, t, s, c)$ or $belief(P, O, V, T, S, C)$. Instead of talking about a predicate P which holds for attribute A with value V, we now describe beliefs as a determinable property P which holds for object O with value V.

Another difference between the current and previous chapter is the use of the two-place predicate $holds\_at$. In Section 3.2 we used this predicate to implement 'memory', for which no difference was made between LTM and STM/WM. In contrast, in Section 4.2 of the current chapter we use the $holds\_at(b, t)$ predicate to denote that belief b held at time t in STM.

In addition, due to the development of a memory model that incorporates availability-values of beliefs, we decided for Section 4.3 to state that a belief b does not hold, but is active at a specific time in STM/WM: $active\_at(b, t)$. In addition, the structure that denotes that a belief is active is in this section referred to as WM instead of STM, although nothing changed in the way it operates.

Research Paper

4.2 A Formal Approach to Aggregated Belief Formation

Abstract

This paper introduces a formal method to aggregate over basic beliefs, in order to deduce aggregated or complex beliefs as often used in applications. Complex beliefs can represent several things, such as a belief about a period in which other beliefs held or the minimal or maximal certainty with which a belief held. As such they contain richer information than the basic beliefs they are aggregated from and can be used to optimize an agent's search through its memory and its reasoning processes. The developed method can also aggregate over aggregated beliefs, hence nested aggregations are possible. An implementation in Prolog demonstrates its operationality.

This section is published as:
Heuvelink, A., Klein, M. C. A., and Treur, J.* A Formal Approach to Aggregated Belief Formation. In M. Klusch, M. Pechoucek, and A. Polleres, (Eds.), *Proceedings of the 12th International Workshop on Cooperative Information Agents* (CIA 2008), p. 71-85, Lecture Notes in Artificial Intelligence, vol. 5180, Springer Verlag. September 10-12 2008, Prague - Czech Republic.
* Authors are listed in alphabetic order and can be regarded as having made a comparable contribution.

4.2.1 Introduction

Agents in applications commonly store beliefs about the state of the world in what is often called a world model or belief base. This belief base is usually a set of atomic beliefs that grows over time. There are several potential problematic issues related to such a belief base. First, after some time the size of the belief base can result in the practical problem that retrieval and inferences will become time expensive: the time needed will grow with the size of the belief base. Second, some inferences result in intermediate results that might be useful again at a later point in time. When the intermediate results are not stored, they have to be recalculated, while when they are stored, they will add to the size problem. Third, this way of storing beliefs seems not very similar to the human way of using memory. For example, humans often forget specific details, but still remember aggregated abstractions or consequences of specific facts. Taking this as a point of departure, it may be explored how aggregated beliefs can be formed and stored within an agent as new entities.

Fact is that aggregations can be formed from many different perspectives and at multiple levels. Which perspectives are chosen and which level of aggregation is needed, is application and task dependent. Therefore a general approach that distinguishes all possible types of aggregations one by one, may become quite complex; for example, if m different aggregation types are possible, and n levels of aggregation, then the number of aggregation types is m^n, which already for relatively low numbers such as $m = 10$ and $n = 5$ leads to a high number (100.000) of aggregated beliefs. To avoid this explosion, in this paper an algebraic approach is adopted that distinguishes a general notion of aggregation operator that (1) is parameterized by the specific constraint that is used in an aggregation process, and (2) can be used in a recursive manner. Thus the combinatorics induced by different levels is replaced by term expressions that can be formed by nesting a number of (parameterized) variants of the aggregation operator.

The presented formalism allows for the specification of complex beliefs at a higher level of aggregation than the basic atomic beliefs. Such aggregated belief representations have the advantage that they are often closer to the level of aggregation that is used in specification of reasoning steps, and are therefore often more useful. For example, it is more convenient to specify reasoning steps based on an aggregated belief such as the 'last most certain belief', than based on a long list of atomic beliefs at different time points and with different certainties. Moreover, some aggregations are used several times. In this case, the aggregation functions as a reasoning template that specifies how a new belief can be deduced from other beliefs. This template only has to be specified once and can be reused later on.

The remainder of the paper is structured as follows. First, Section 4.2.2 introduces an

application domain and the basic belief formalism that is used. In the subsequent section, the algebraic approach to belief aggregation is described, which is formalized as a term algebra in Section 4.2.4. Section 4.2.5 and 4.2.6 demonstrate the operationality of the approach, by presenting a Prolog implementation and showing how the algebra allows the formation of several useful complex beliefs, which are defined as specific aggregations. Section 4.2.7 relates the work to other research, both in the area of knowledge compilation and temporal abstraction. Finally, in Section 4.2.8 the research is summarized and future research plans exposed.

4.2.2 Belief Formalism

In (Heuvelink and Both, 2007) a software agent was developed that compiles a tactical picture of its environment, which entails the classification and identification of surface radar contacts. For modeling its behavior the need was identified to explicitly represent the time at which a belief was held by the agent in its short term memory (STM). In other words: when it believed it. The main reason to represent this, was to enable the (biased) reasoning over (possible inconsistent) beliefs over time (Heuvelink and Both, 2007). For example, when at time t it is believed that the position of a contact is [x1, y1], while at time t + n it is believed to be position [x2, y2], the average speed of the contact can be inferred. This is useful because the speed on a contact might contain information concerning its identity, e.g., large ships that are neutral usually do not sail faster than 20 knots. In the same way a contact's maneuvering pattern can be inferred, which is relevant as it gives away much information concerning a contact's intent.

In order to logically represent other aspects, namely uncertainty of information, and the fact that information can come from various sources, every belief also received a source and a certainty label. As a result, the basic knowledge entity of the agent is represented by $belief(P, O, V, T, S, C)$, which denotes the belief that the indeterminable property P holds for object O with the value V at time T, based on source S and with certainty C. An example belief denoting that it is believed at time 8 with high certainty that the identity of the radar contact1 is friendly because of radio conservation is: $belief(identity, contact1, friendly, 8, radio, 0.9)$.

The value, time, and certainty label of beliefs about a specific property and object are often used to reason about trends in those beliefs, which can lead to new beliefs. For example, a new belief can be formed about a contact being a merchant, and therefore neutral, due to beliefs about it sailing in a straight line. The certainty of the belief that the contact is a merchant is determined by the period, as by the certainty, with which it is believed that it does this. For the deduction of other beliefs it is often important to deduce what the last, or most (un)certain belief about a specific something was. For example,

the highest certainty with which it was once believed that a contact fired is relevant for deducing whether it might be hostile.

The beliefs formed by the agent over time are stored in the agent's belief base, representing long term memory (LTM). When storing beliefs in LTM, it is important to denote when they were formed or retrieved in STM. For this a new reference to time is introduced, the two-place predicate $holds\_at$. When the basic belief predicate of the object language is reified to a term b, the time at which the belief is held in STM can be expressed by $holds\_at(b, t)$. For every $belief(p, o, v, t, s, c)$ that is found in the agent's belief base it holds that $holds\_at(belief(p, o, v, t, s, c), t)$, since the t of the belief denotes the time it was formed (was present in STM).

4.2.3 Belief Aggregation

Unfortunately, the storage of time-stamped beliefs led to the problematic issues mention in section 4.2.1 (Both and Heuvelink, 2007). Therefore, this paper focuses on the development of a generic, formal approach to the formation of arbitrary aggregations over these basic beliefs, to form all kinds of so-called aggregated or complex beliefs. Complex beliefs abstract or cluster information of the lower level. They form a solution for keeping the amount of time required to search through the agent's belief base within limits. Furthermore, they can be used to model specific properties of human memory, like the forgetting of specific details.

Aggregation Examples

An example of a complex belief that an agent can form was mentioned in Section 4.2.2, namely a belief about the period during which a certain belief held. That complex belief, about the duration of the straight maneuvre of a contact, can be used directly to infer a new belief, e.g., that that contact might be a merchant.

While specifying the formal model underlying the reasoning and behavior of the cognitive agent described in (Heuvelink and Both, 2007), it was found that often specific types of information are required to deduce new beliefs. To be precise, often the last, earliest, most certain or uncertain, increasing in certainty, or longest held belief was required. In addition, it was noticed that the deduction processes of several of these beliefs are very similar, e.g. the deduction of a last belief (belief with highest T) is very similar to the deduction of a most-certain belief (belief with highest C).

These observations spurred the development of an generic approach to belief aggregation in which a complex belief is defined as an aggregation that takes the form of a constraint (e.g., highest) that must hold for a certain variable (e.g., T) of a certain more

of less specified belief (e.g., belief(identity, contact1, friendly, T, S, C) in which the P, O, and V are specified while the T, S, and C are left variable). The term algebra formalizing this approach to the formation of aggregated beliefs is introduced in Section 4.2.4, while the section after that discusses an implementation of the approach in Prolog.

Complex Belief of Type Integrated Sources

The $integrated\_sources$ belief was the most important complex belief the developed agent in (Heuvelink and Both, 2007) reasoned with instead of with its basic beliefs. This complex belief represents which value is currently believed by the agent to hold for a certain property and object, and with which certainty. To determine this, inconsistencies formed by beliefs from different sources, with different certainties, and held at different times, have to be resolved. Much research has been done on how to deal with such inconsistencies, (see, e.g., Castelfranchi, 1997; Bloch et al., 2001).

In (Heuvelink and Both, 2007) a relative simple procedure was introduced to determine which value V was currently believed to held with certainty C by an agent for a given P and O. This procedure takes into account that a belief's validity over time is strongly influenced by its predicate type (property) P. Values of some predicates are much more persistent than others; consider the chance that a contact's position, speed, or intent changes over time. The following logical expression denotes the meaning of the complex belief called $integrated\_sources$:

$$
\begin{array}{l}
given(p, o) \\
\forall v1\ \forall t1\ \forall s1\ \forall c1\ \forall t\ \forall pd\ [\\
\quad holds\_at(complex\_belief(integrated\_sources, for(p, o), has\_values(v1, c1 - pd * (t - t1))), t) \\
\quad \leftrightarrow \\
\quad holds\_at(complex\_belief(last, for(p, o, s1), has\_values(v1, t1, c1)), t) \wedge \\
\quad holds\_at(persistence\_decay(p, pd), t) \wedge \\
\quad \neg\exists s2\ \exists v2\ \exists t2\ \exists c2\ [\\
\qquad holds\_at(complex\_belief(last, for(p, o, s2), has\_values(v2, t2, c2)), t) \wedge \\
\qquad c2 - pd \times (t - t2) > c1 - pd \times (t - t1)\ ]\ ]
\end{array}
$$

This expression specifies that the agent believes at time t that for a given P and O, $for(p, o)$, the value $v1$ holds, which is the value of the belief about P and O whose certainty is the greatest after taking into account the time passed since it was formed and the persistence of the property; $c1 - pd * (t - t1)$. This might entail that the value of an older belief with a certain certainty is believed over the value of a newer belief that has a lower certainty. It might also be the other way around; it depends on the nature (persistence) of the property. In this expression another complex belief was used of the type $last$, which has as exact definition:

$$
\begin{aligned}
&\forall p\, \forall o\, \forall v\, \forall t\, \forall s\, \forall c\, \forall n\, [\\
&\quad holds\_at(complex\_belief(last, for(p,o,s), has\_values(v,t,c)), n)\\
&\quad \leftrightarrow\\
&\quad [\, holds_a t(belief(p,o,v,t,s,c), t)\ \wedge\ t \leq n\, \wedge\\
&\quad\ \neg\exists t'\, \exists v'\, \exists c'\, [\\
&\qquad holds_a t(belief(p,o,v',t',s,c'), t'\ \wedge\ t' \geq t\ \wedge\ t' \leq n\,]\,]\,]
\end{aligned}
$$

This last expression specifies that the agent believes at time n that t is the last time at which a belief incorporating the given P, O, and S, $for(p, o, s)$, held. This is the case since t is the time label of a belief with that given P, O, and S, for which it holds that no other belief exists with the same P, O, and S, but a higher T (t'). This complex belief of type last is defined as an aggregation of all the beliefs with the given P, O, and S, and the constraint Highest for their time label T. Besides a specification for T, this aggregation also specifies the free variables V and C. This is a quite standard aggregation, considering the limited complexity of the constraint that it takes into account. The aggregation as which the complex belief of type $integrated\_sources$ is defined, is much less standard. The constraint that has to be taken into account in that aggregation is much more specific and not likely to be reusable, see Section 4.2.5.

The current paper focuses on the development of an algebraic approach to have an efficient representation of aggregated beliefs. For demonstration purposes it elaborates on several possible types of these aggregated beliefs, which are defined as specific aggregations. Notice that the introduced aggregations simply serve as examples, and that many more are possible. The approach is set up in such a generic way that all kinds of constraints that lead to all kinds of complex beliefs can be expressed with it.

4.2.4 Algebraic Formalization

The algebra specification of the aggregation functions on beliefs is defined by a basic ontology, by means of which its objects and relations can be expressed. The primitive terms used in the algebra are defined by a many-sorted signature. The signature takes into account symbols for *sorts*, *constants*, *functions* and *relations*. Examples of *sorts* are: LABEL, CONSTRAINT, TIME, TYPE, AGGREGATIONBASE, AGGREGATEDBELIEF, ARGUMENTLIST, BASICBELIEFBASE, PROPERTY, or OBJECT. *Constants* are names of objects within sorts; examples are 'speed', '20', or 'fast'. *Functions* denote mappings from a (combination of) sort(s) to another sort; examples of function symbols are agg, $holdsat$, $+$ and $*$. *Relations* symbols (relating different sorts) used are, for example $=$ and $<$. Logical relationships involve conditional statements involving relations. Figure 4.1 depicts a large part or the algebra specification with the definitions used listed below. Arrows with no label are defined by e which denotes (injective) embedding.

agg: LABEL x CONSTRAINT x AGGREGATEDBELIEF → AGGREGATIONNAME
e: AGGREGATIONBASE → AGGREGATEDBELIEF
e: BASICBELIEFBASE → AGGREGATEDBELIEF
e: COMPLEXBELIEFBASE → AGGREGATEDBELIEF
$holdsat$: AGGREGATIONNAME x TIME → AGGREGATIONBASE
$definedas$: COMPLEXBELIEFBASE x AGGREGATIONBASE
$holdsat$: COMPLEXBELIEF x TIME → COMPLEXBELIEFBASE
$complexbelief$: TYPE x ARGUMENTLIST x RANGELIST → COMPLEXBELIEF
e: PROPERTY → ARGUMENTLIST
e: OBJECT → ARGUMENTLIST
e: VALUE → ARGUMENTLIST
e: TIME → ARGUMENTLIST
e: SOURCE → ARGUMENTLIST
e: CERTAINTY → ARGUMENTLIST
e: RANGE → RANGELIST
$holdsat$: BASICBELIEF x TIME → BASICBELIEFBASE
$belief$: PROPERTY x OBJECT x VALUE x TIME x SOURCE x CERTAINTY → BASICBELIEF
$abstraction1$: LABELTYPE x VAR → LABEL
$abstraction2$: LABELTYPE x LABELTYPE x VAR x VAR → LABEL
$constraint$: NAME x VARIABLE x AGGREGATEDBELIEF → CONSTRAINT
$forall, exists$: VAR x FORMULA → FORMULA
$definedas$: CONSTRAINT x FORMULA
$<, >, \leq, \geq$: TERM x TERM → ATOM
e: ATOM → FORMULA
not: FORMULA → FORMULA
$and, or, implies$: FORMULA x FORMULA → FORMULA

A number of sorts are considered primitive; they only contain constants such as names and values: LABELTYPE, VARIABLE, PROPERTY, OBJECT, VALUE, TIME, SOURCE, CERTAINTY, RANGE, and TYPE. Some other sorts are more or less standard, and/or may depend on application dependent functions: ATOM, TERM, FORMULA.

Sort ARGUMENTLIST 1 contains terms listing 6 arguments with at each of the 6 positions instances. Two special instances exist; $free$ and $range$, which denote that the argument of that position is variable. The sort RANGELIST contains terms listing the 6 ranges for the 6 arguments of ARGUMENTLIST 1. The range is only relevant for the arguments with the special instance $range$. In the case of a normal instance the corresponding range is nr (not relevant) while in the case of the special instance free, the corresponding term is any. Sort ARGUMENTLIST 2 contains terms listing 6 arguments with at each of the 6 positions instances. Two special instances exist: $given$ and nr,

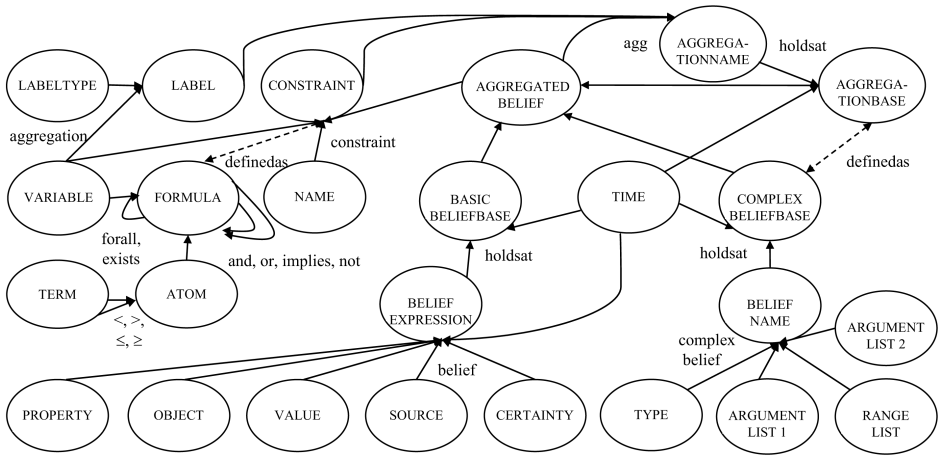


Figure 4.1: Overview of the algebra for belief aggregation

which denote respectively that the argument of that position was already specified in ARGUMENTLIST 1, or is no longer relevant given the TYPE.

Note that the function agg can be used in a recursive manner together with the function $holdsat$. The nested term structures that result, represent beliefs at different levels of aggregation: the level is the number of nested agg functions occurring in the term.

The area of algebraic specification has a long history. From the extensive literature techniques can be borrowed to obtain an implementation of calculations in the algebra, for example in a functional or logic programming language. If relations are involved, an implementation has to take into account both functional and logical aspects; (e.g., Drosten, 1988; Hanus, 1994). Following this tradition, the next section introduces an implementation of the developed algebra in the logic programming language Prolog.

The algebra is considered a term algebra, which specifies the different variations of aggregated belief expressions that can be formed. The current Prolog implementation generates such expressions, but does not perform evaluations of whether two different expressions should be considered as having the same content or meaning. In future work it is planned to extend this approach to an algebra for which also equations are specified, and an implementation where such equations are incorporated.

4.2.5 Implementation

The algebra of section 4.2.4 is implemented in SWI-Prolog (Wielemaker, 2003). In this section, the implementation choices are explained. For the readability of this section, only parts of the Prolog program are shown. The complete source code can be down-

loaded from: http://human-ambience.few.vu.nl/docs/CIA-AggregationAlgebra.pl.

Controlling Aggregations

The current implementation does not incorporate automatic control of aggregations. Instead two 'programs' are implemented that can be called from the Prolog-shell: `holds` and `post`. The holds program is shown below and can be used to request the results of a specific aggregation, or to request the values for which a specific complex belief holds. Notice that complex beliefs are defined as aggregations and are as such interchangeable. When no `holds_at` attribute is included in the query, it is assumed that the query requests the result of the aggregation or complex belief at the current time. When a `holds_at` is included, the query requests the result of the aggregation or complex belief that holds at the specified time.

`holds(B):-`
`  B = complex_belief(_,_,_,_),`
`  current_time(N),`
`  complex_belief_is_defined_as(`
`  holds_at(B,N),`
`  holds_at(agg(L,C,A),N)),`
`  holds_at(agg(L,C,A),N).`
	Query about B,
	B is a complex belief,
	and is checked for the current time N.
	The definition of the
	complex belief B is
	the aggregation agg(L,C,A),
	which is requested for time N.

`holds(B):-`
`  B = holds_at(X,N),`
`  X = complex_belief(_,_,_,_),`
`  complex_belief_is_defined_as(`
`  B, holds_at(agg(L,C,A),N)),`
`  is_time(N),`
`  holds_at(agg(L,C,A),N).`
	Query about B,
	B is whether X holds at time N,
	with X being a complex belief.
	The complex belief X within B
	is defined as an aggregation.
	When N is an actual time,
	that aggregation is requested for N.

`holds(B):-`
`  B = agg(L,C,A),`
`  current_time(N),`
`  holds_at(agg(L,C,A),N).`
	Query about B,
	B is an aggregation,
	and is checked for the current time N,
	and therefore requested at N.

`holds(B):-`
`  B = holds_at(agg(L,C,A),N),`
`  is_time(N),`
`  holds_at(agg(L,C,A),N).`
	Query about B,
	B is whether agg(L,C,A) holds at
	N, when this is an actual time
	agg(L,C,A) is requested at N.

The `holds` program does not alter the belief-base and as such can be used to investigate 'what-if' questions. Besides the `holds` program also a `post` program exists whose procedure is almost identical, except each of its rules is extended with the extra condition `assert(_)`. This adds the complex belief, defined as the aggregation that is checked to hold at N, to the belief base.

Free and Bounded Variables

Complex beliefs do not have 6 atomic arguments like basic beliefs, but are made up of four arguments. The first of these is atomic and specifies a name for the complex belief, which is also referred to as its *type*. The latter three arguments are predicates; each embeds 6 arguments whose positions respectively represent the P, O, V, T, S, and C. So a complex belief is represented by `complex_belief(Type, For(...), With_Ranges(...), Has_Values(...))` , which denotes that it is believed by the agent that for that complex belief `Type` and for the given constants in `For` , taken into account the `With_Ranges` in which the free variables in `For` have to lie, the constants in `Has_Values` count.

An example complex belief denoting that it is believed that the last time at which a belief was held about the hostile identity of contact1 is 7, and that was with a certainty 0.6 and based on radio contact is:

```
complex_belief(last,
    for(identity, contact1, hostile, free, free, free),
    with_ranges(nr, nr, nr, any, any, any),
    has_values(given, given, given, 7, radio, 0.6)).
```

Such a complex belief is the result of the agent reasoning about what the last time, i.e. highest T, was that it believed that the identity of contact1 was hostile. When it would have reasoned about what the last time was that it believed with less than 0.5 certainty that that was the case, the following complex belief might have hold:

```
complex_belief(last,
    for(identity, contact1, hostile, free, free, range),
    with_ranges(nr, nr, nr, any, any, [0, 0.5]),
    has_values(given, given, given, 4, vision, 0.4)).
```

Given this representation of complex beliefs, an example of a `complex_belief_is_defined_as` relation which defines a complex belief as a specific aggregation is:

```
complex_belief_is_defined_as(
    holds_at( complex_belief(
        last,
        for(P,O,V,free,S,C),
        with_ranges(nr,nr,nr,any,nr,nr),
        has_values(given,given,given,X,given,given)),N),
    holds_at( agg( temporal_aggregation(T),
                   highest_free(X,any,P,O,V,S,C),
                   holds_at(belief(P,O,V,T,S,C),N)),N)).
```

The aggregation shown here will return the highest T that it can find for the given P, O, V, S, and C. When it does not matter what the S and C are, but it is required to find the highest (last) T that is now restrained to a certain time range [Tb, Te] for a given P, O, and V, the following aggregation is applicable:

```
complex_belief_is_defined_as(
   holds_at( complex_belief(
      last,
      for(P,O,V,range,free,free),
      with_ranges(nr,nr,nr,[Tb,Te],any, any),
      has_values(given,given,given,X,Y,Z)),N),
   holds_at( agg(
      temporal_source_certainty_aggregation(T,S,C),
      highest_range_free_free(X,[Tb,Te],Y,any,Z,any,P,O,V),
      holds_at(belief(P,O,V,T,S,C),N)),N)).
```

This aggregation will return the highest T that it can find for the given P, O, and V. The S and C that it returns are those of the belief with that highest T. This aggregation example demonstrates that variables can be free or that they can be restricted to a specific range. When it is checked whether a certain aggregation holds at a certain time the following clause executes:

`holds_at(agg(L,C,A),_):-`
`  term_variables(L,V),`
`  constraint_is_defined_as`
`  (constraint(C,A,V),F),`
`  F.`
	To determine the agg(L,C,A) at time _, the variables in L are listed in V. The definition of the constraint C for the subject A and the variables in V is F, which is consequently requested.

The first condition term_variables(L,V) is a built-in Prolog predicate that unifies V with a list of variables, each corresponding with a unique variable of L and ordered in order of appearance in L. So for the example above it holds:

```
?- term_variables(temporal_certainty_source_aggregation(T,C,S),V).
V=[G34,G35,G27], T=G34, C=G35, S=G27.
```

The second condition is a user-defined predicate that defines what the constraint C entails for the aggregated belief A with its free variables listed in V; namely F, which forms the last condition. On the next page, an example `constraint_is_defined_as` is shown for the constraint that is required to deduce the first complex belief of type `last` introduced in this section. Notice that this `highest_free` constraint can be reused, e.g., to deduce a complex belief of type `surest` when it is combined with a `certainty_aggregation` . Its logical expression is:

$given\ A, \forall x\ [\ highest\_free(x, any)\ \leftrightarrow\ A(x)\ \wedge\ \forall x1\ [\ A(x1)\ \rightarrow\ x1 \leq x\ ]\ ]$

```
constraint_is_defined_as(
constraint(
  highest_free(X,any,F1,F2,F3,F4,F5),
  A1, [X1]),
and( copy_term( (A1,X1,F1,F2,F3,F4,F5),
                (A,X,F1,F2,F3,F4,F5)),
  and(A, forall(A1, X1=<X)))).
```

Definition (
constraint (
C,
A, V),

F).

The constraint that was required to deduce the second complex belief of type `last` introduced in this section, is shown next. It can be seen that this constraint only considers options A1 whose values X1 for the variable X lies within the range [Xb, Xe] specified for it.

```
constraint_is_defined_as(
constraint(
  highest_range_free_free(
    X,[Xb,Xe],Y,any,Z,any,F1,F2,F3),
  A1, [X1, Y1, Z1]),
and( copy_term( (A1,X1,Y1,Z1,F1,F2,F3),
                (A,X,Y,Z,F1,F2,F3)),
  and(A, and( X>=Xb, and( X<Xe,
    forall( and(A1, and( X1>=Xb, X1<Xe)),
            X1=<X))))))).
```

Definition (
constraint (
C,
A, V),

F).

Nested Aggregations

The reason that in the `constraint_is_defined_as` Prolog clauses the values F1, ..., Fn are embedded is that although they are usually instantiated, they do not have to be. When they are not, and are left out of the query, they do get instantiated when Prolog requests A. However, when next is asked whether for all X1 in A X1 $\leq$ X holds, this probably fails. This is because the left-out variable that now is instantiated in A, is still free in A1, so much more A1's are checked than there should be.

The reason why variables are allowed to exist in places where atoms are expected is because this freedom enables nested aggregations. An example of a nested aggregation is the complex belief `integrated_sources` introduced in Section 4.2.3:

```
complex_belief_is_defined_as(
   holds_at( complex_belief(
      integrated_sources,
      for(P,O,free,free,free,free),
      with_ranges(nr,nr,any,any,any,any),
      has_values(given,given,X,nr,nr,Y)),N),
```

```
holds_at( agg(
    certainty_temporal_source_value_aggregation(C,T,S,V),
    highest_free_after_free_for_free_free_for_predicate_&_time
                (Y,any,_,any,_,any,X,any,P,O,N),
    holds_at(complex_belief(
                last,
                for(P,O,free,free,S,free),
                with_ranges(nr,nr,any,any,nr,any),
                has_values(given,given,V,T,given,C)),N)),N) )
```

In this clause a complex belief of type `last` functions as aggregated belief for the aggregation that deduces the complex belief of type `integrated_sources` . This latter aggregation aggregates over values, times, sources, and certainties of beliefs about a given property and object, in order to retrieve a specific value and certainty. The aggregation belief it needs as input is a complex belief of type `last` that aggregates over values, times and certainties for a given property, object and source. However, the latter (S) is not given but variable, because the top-aggregation needs this `last` type for all possible sources. Note that instead of the complex belief of type `last` also the aggregation as which it is defined could have been used as input.

The constraint used within the aggregation to deduce the complex belief `integrated_sources` is much more specific and therefore less reusable than, e.g., the `highest_free` constraint. These two examples nicely illustrate the reach of the proposed aggregation mechanism. In principle all possible constraints can be added and used to form new types of complex beliefs that in turn can be used in other aggregations.

4.2.6 Example Scenarios

From http://human-ambience.few.vu.nl/docs/CIA-AggregationAlgebra.pl the source code of our Prolog program can be downloaded. In the case presented, an agent attempts to infer the identity of a radar contact. Information about this contact can be gathered by the radar as well as by the agent's own vision. Furthermore, the agent can generate new beliefs by reasoning over other beliefs. Over time the following basic beliefs have held in STM and are now stored in LTM:

```
holds_at(belief(identity,contact1,neutral,2,vision,0.5),2).
holds_at(belief(identity,contact1,neutral,3,radar,0.3),3).
holds_at(belief(speed,contact1,20,3,radar,0.9),3).
holds_at(belief(identity,contact1,hostile,4,vision,0.9),4).
holds_at(belief(identity,contact1,hostile,4,id_from_speed,0.4),4).
holds_at(belief(speed,contact1,28,6,radar,0.9),6).
```

```
holds_at(belief(speed,contact1,30,7,vision,0.5),7).
holds_at(belief(identity,contact1,hostile,8,vision,0.7),8).
holds_at(belief(identity,contact1,hostile,8,id_from_speed,0.8),8).
```

At current_time 10, two of the nine beliefs stored in the agent's LTM are formed by the agent's reasoning rule `id_from_speed`, which forms the source of those beliefs. At this moment the agent might start another reasoning process for which it requires the last belief about a hostile identity of contact1. This query results in:

```
1 ?- holds(complex_belief(last, for(identity,contact1,hostile,
free,free,free), with_ranges(nr,nr,nr,any,any,any), has_values
(given,given,given,T,S,C))).
T = 8,
S = vision,
C = 0.7 ;
T = 8,
S = id_from_speed,
C = 0.8 ;
fail.
```

By chance, two sets of atoms are found that both adhere to this query. In such case the agent might be interested in the surest one of these two last beliefs. This complex belief of type `surest_last` is formed by aggregating the label `certainty_temporal_source_aggregation` and the `highest_free_free_free` constraint with that complex belief of type `last` as aggregated belief. The query for complex belief of type `last_surest` yields a totally different result: it is an aggregation of the same constraint but in combination with a `temporal_certainty_source_aggregation` and on complex beliefs of the type `surest`.

```
2 ?- holds(complex_belief(surest_last, for(identity,contact1
,hostile,free,free,free), with_ranges(nr,nr,nr,any,any,any),
 has_values(given,given,given,T,S,C))).
T = 8,
S = id_from_speed,
C = 0.8 ;
fail.

3 ?- holds(complex_belief(last_surest, for(identity,contact1
,hostile,free,free,free), with_ranges(nr,nr,nr,any,any,any),
 has_values(given,given,given,T,S,C))).
T = 4,
S = vision,
C = 0.9 ;
fail.
```

Another possibility would be that the agent's superior asks the agent what it believes that contact1's identity is. At that moment the agent will retrieve its last beliefs about the identity of that contact and form an answer. In the current case the agent believes contact1 might be neutral based on what it saw of the vessel, as on the radar-emission-

pattern it received from the contact. However, it also believes it might be hostile, due to its high speed. In order to give its superior an answer the agent has to form a belief about the contact's identity by integrating the retrieved last information about its identity from the different sources. Given that the persistence-decay of a contact's identity (see Section 4.2.3) is 0, the agent reports it believes the contact to be hostile since it was most sure of that. The agent's superior could also have asked what the agent believes that

```
4 ?- holds(complex_belief(integrated_sources, for(identity
,contact1,free,free,free,free), with_ranges(nr,nr,any,any,
any,any), has_values(given,given,V,nr,nr,C))).
V = hostile,
C = 0.8 ;
fail.
```

the speed of contact1 is. Again the agent needs to integrate information from different sources and times. However, because the persistence-decay of speed is larger than 0, say 0.05, it also has to take into account how long ago it was that it believed that information. The answer it will give is 28, see below. This knowledge is deduced from the basic belief at time 6 that its speed was 28, but notice that the certainty with which it is believed has decayed; from 0.9 to 0.7. Moreover, a newer belief concerning the contact's speed existed. However, even though the predicate's certainty decreases over time, still the value of the older belief is believed because the certainty of the new belief was very low.

```
5 ?- holds(complex_belief(integrated_sources, for(speed,
contact1,free,free,free,free), with_ranges(nr,nr,any,any
,any,any), has_values(given,given,V,nr,nr,C))).
V = 28,
C = 0.7 ;
fail.
```

4.2.7 Related Research

The technique for pre-processing a knowledge base to derive intermediate conclusions that is presented in this paper is related to the area of knowledge compilation. Knowledge compilation is defined in (Cadoli and Donini, 1997) as 'methods of processing off-line a knowledge base in such a way that the output of such a pre-processing can be used to speed up on-line answering for a class of queries, where the pre-processing should take an finite amount of time'. Within the area of knowledge compilation a distinction is made between exact methods (which are sound and complete) and approximate methods, which either reduce the complexity by expressing the knowledge or query in a simpler language or by leaving out some (complex) parts of the knowledge base.

Our approach is an exact technique, as it only results in sound intermediate results. However, a difference with common techniques for knowledge compilation is that our method does not strive to derive all intermediate results, whereas knowledge compilation techniques usually try to find a representation of all theorems of the initial knowledge base. For example, they transform a knowledge base to normal form and compute all implicants or implicates. In contrast, our approach is driven by specific queries whose results are likely to be useful for the task execution. In that sense, our method is not complete, as it does not aim to represent all knowledge in a different representation. Moreover, our aggregations are usually more complex (and thus richer in information), whereas knowledge compilation techniques often result in simpler representations. Last, our aim is to compile new knowledge on-line instead of off-line.

Shahar (1997) presents a framework for knowledge based temporal abstraction from time-stamped data. His formal specification of a domain's temporal-abstraction knowledge supports acquisition, maintenance, reuse, and sharing of that knowledge. His aim is partly the same as our, however, his framework allows for temporal abstractions only, whereas our algebra allows for arbitrary abstractions.

The area of belief revision is also related to our work. In belief revision, the question is how existing beliefs are influenced when new pieces of information are taken into account, for example when information is added, removed or changed. The dominant theory on belief revision, the so-called AGM model (Alchourròn et al., 1985), formulates properties that an operator that performs revision should satisfy in order for being considered rational. Similarly, related work on belief merging focuses on the consequences of combining belief bases for the integrity of a belief base, for example, see (Konieczny and Pino Prez, 2002). In our work the logical consequences of the aggregations are not relevant, as no new knowledge is added. There is no inconsistent information that is merged and it does not happen that old information changes because all information is time-stamped. This is comparable to what Sripada (1993) describes, who also uses time-stamped beliefs. He proposes a technique for the efficient revision of beliefs in knowledge bases for real-time applications, but only looked at binary beliefs.

Another type of related work is formed by approaches for memory storage in existing (cognitive) agent architectures. In a recent review study on computer-based human behavior representations (Morrison, 2003) it was generalized that 'all the (human behavior) models can represent either short term memory (STM) or long term memory (LTM).' However, the ways in which these memories function differ greatly. For example, ACT-R's STM is formed by a retrieval buffer that can hold one chunk, which it retrieves using an activation function from its declarative memory module (LTM) (Anderson and Lebiere, 1998), while Soar's STM is formed by its working memory that is not limited

in the number of elements it can hold (Laird et al., 1987). Related to the differences in memories, differences exist in the representation of the declarative information entities stored in such modules. These representations range from nodes in a network with an activation value to first-order propositions.

The functioning of the various memories are in general fixed and tuned to bring about the behavior for which the architecture was developed. No existing architecture is build to specifically deal with time-labeled constructs, let alone in the algebraic approach as introduced in this paper. Despite this, it might be possible to map the specific belief construct to the memory construct of an architecture, prohibited the form of the latter has a certain degree of freedom (Both and Heuvelink, 2007; Muller et al., 2008). Moreover, the constraints that are needed to infer required (possibly domain-specific) aggregations have to be implemented in the architecture as well, as the aggregation algebra.

4.2.8 Summary and Future Research

In this paper a method and a term algebra is presented to form arbitrary aggregations of beliefs in a knowledge base. The aggregations can be formed at different levels and from different perspectives, i.e. time aggregations, source aggregations, certainty aggregations, etc. A Prolog program is used to illustrate the feasibility of the approach. The motivation of this work is twofold: it should help to improve the computational problems when reasoning over a knowledge base, and it should reflect a more human way of storing information in memory. As such, the goal is to 'validly' represent aggregations of humans over beliefs, both conscious as subconscious, which can be used in agent applications where agents should behave in a human-like way.

Up to now, the control of the formation of aggregations is not yet implemented. Future research will investigate the control of aggregations from two perspectives. The first will be inspired by the human processes of forgetting and remembering, the second by the human processes of attention and focusing in task execution.

Research Paper

4.3 An Agent Memory Model Enabling Rational and Biased Reasoning

Abstract

This paper presents an architecture for a memory model that facilitates versatile reasoning mechanisms over the beliefs stored in an agent's belief base. Based on an approach for belief aggregation, a model is introduced for controlling both the formation of abstract and complex beliefs and the retrieval of them based on their activation history. An implementation of the presented mechanisms illustrates how it can be used in intelligent agents that exhibit human-like (biased) reasoning as well as rational reasoning.

This section is published as:
Heuvelink, A., Klein, M. C. A., and Treur, J.* An Agent Memory Model Enabling Rational and Biased Reasoning. In *Proceedings of the Seventh IEEE/WIC/ACM International Conference on Intelligent Agent Technology* (IAT 2008), p 193 - 1999, IEEE Computer Society Press. December 9 - 12 2008, Sydney, Australia.
* Authors are listed in alphabetic order and can be regarded as having made a comparable contribution.

4.3.1 Introduction

Agents commonly store beliefs about the state of the world in what is often called a world model or belief base that grows over time. Reasoning can be done by applying inference rules to the stored beliefs. This way of storing and using beliefs seems not very similar to the human way of using memory. For example, humans often forget specific details, but still remember aggregated details or abstractions of specific facts. Therefore, human reasoning is not always sound and rational, but sometimes heuristically or biased, based on abstractions or generalizations that are (more easily) available.

A notion describing the human way of using memory is 'episodic memory' (Tulving, 2002). This refers to the memory of events, times, places, associated emotions, etcetera in relation to an experience, which is formed automatically. Episodic memory is contrasted to semantic memory, which is the memory of factual or conceptual knowledge. It is sometimes believed that episodic memories are converted into semantic memories over time. In this process, most knowledge about the specific event is generalized, and abstractions are stored in the semantic memory. Nuxoll and Laird (2004) distinguish 4 phases in episodic memory, namely encoding, storage, retrieval, and use of the retrieved memories.

Taking these ideas about human memory as a point of departure, it has been explored how an agent may form, store, retrieve, and use aggregated beliefs, which can be abstract or complex, see Section 4.3.2. Which perspective and level of aggregation is needed, is application and task dependent. For that reason, an algebraic approach was developed in (Heuvelink et al., 2008b) that can form arbitrary aggregations over beliefs. For this, it uses a general aggregation operator parameterized by a specific constraint that can be used in a recursive manner. The formalism allows for the specification of complex beliefs, e.g., a 'last most certain belief'.

In this paper, a memory model is proposed that also controls the formation and use of these aggregations. The control mechanism it embeds has a hybrid nature: abstract beliefs are automatically generated, while complex beliefs are only generated if the agent has a specific focus, induced by the task of the agent. Retrieval of the various belief types is based on their availability, which depends on their activation history, and on the task focus of the agent.

The model builds on the long-standing idea that forgetting is the key to proper functioning of memory (James, 1890). Recently, Schooler and Hertwig (2005) performed simulation experiments that demonstrate that forgetting might facilitate human inference performance. To implement forgetting in a computational model, beliefs usually get an availability value that decays over time. However, it is human-like that although specific details cannot be remembered, abstractions of beliefs can. Such beliefs are more avail-

able than the more detailed beliefs they are abstractions of, especially in case of multiple detailed instances. This feature of memory is taken into account by the approach introduced.

4.3.2 Memory Model Concepts

This section introduces the memory model predicates that bring about the aspects mentioned in 4.3.1.

Basic Belief Type

The need for a memory model as proposed in this paper was identified in the context of military training simulation (Heuvelink and Both, 2007). An example agent task is the formation of a tactical picture of the environment, which entails the classification and identification of surface radar contacts. For modeling its behavior it is needed to explicitly represent the time at which a belief was actually believed by the agent, to enable (biased) reasoning over (possible inconsistent) beliefs over time. For example, when at time t it is believed that the position of a contact is [x1, y1], while at time t + n it is believed to be position [x2, y2], the average speed of the contact can be inferred. This is useful because the speed may contain information concerning its identity, e.g., large ships that are neutral usually do not sail faster than 20 knots.

In order to logically represent time and other aspects, such as the uncertainty and source of information, every belief receives a number of labels, and is represented by $belief(P, O, V, T, S, C)$. This denotes the belief that property P holds for object O with value V at time T, based on source S and with certainty C. An example belief denoting that it is believed at time 8 with high certainty that the identity of contact1 is friendly because of radio conversation is: $belief(identity, contact1, friendly, 8, radio, 0.9)$. The value, time, and certainty label of basic beliefs about a specific property and object are often used to reason about trends in those beliefs, which can lead to new beliefs. For example, a new belief can be formed about a contact being a merchant, and therefore neutral, due to beliefs about it sailing in a straight line. The certainty of the former is determined by the period, as by the certainty, with which the latter is believed. For the deduction of other beliefs it is often important to deduce the last, or most (un)certain belief about something. For example, the highest certainty with which it was once believed that a contact fired is relevant for deducing whether it might be hostile.

Types and Control of Aggregations

The need was identified for the memory model to support besides the conscious reasoning processes on beliefs, more unconscious processes on beliefs, referred to as aggregations. The approach introduced in (Heuvelink et al., 2008b) allows for the formation of arbitrary aggregations over beliefs, which result in aggregated beliefs. In this approach an aggregated belief is defined as an aggregation that takes the form of a constraint (e.g., $highest$) that must hold for a certain variable (e.g., T) of a partially specified belief (e.g., $belief(identity, contact1, friendly, T, S, C)$ in which the P, O, and V are specified while the T, S, and C are left variable). Examples of aggregated beliefs that an agent can form are mentioned in the previous section, e.g., a belief about the highest certainty with which something was believed.

In this paper two types of aggregations are distinguished. The first aggregates over beliefs with many detailed arguments to form an $abstract\ belief$ that 'forgets' certain details. Such a process is referred to as a $basic\ aggregation$. This process simply abstracts from a belief occurrence by leaving one or more of its arguments free. The basic aggregation process generates an $abstract\_belief(AbstractionType, ForArguments, WithRanges)$ as output and requires a $belief(P, O, V, T, S, C)$, or another $abstract\_belief$ as input.

For example, the basic belief $belief(identity, contact2, hostile, 4, vision, 0.9)$ can be abstracted into:

$$abstract\_belief(certainty\_temporal\_abstraction,\\ for(identity, contact2, hostile, free, vision, free),\\ with\_ranges(nr, nr, nr, any, nr, any))$$

by executing the following aggregation:

$$agg(certainty\_temporal\_aggregation(C, T),\\ abstract\_free\_free(\_, any, \_, any, P, O, V, S),\\ is\_available\_at\_current\_time(belief(P, O, V, T, S, C)))$$

These abstract beliefs are formed in an automatic way, as a feature of memory. However, which abstractions happen and the amount of detail that is abstracted from, might be susceptible to certain internal aspects, e.g., stress. Therefore, it is required that these basic aggregations can be controlled at each time point. For this the predicate $in\_memory\_focus\_at(B, T)$ is used. When a certain belief is in memory focus at a certain time, the aggregation at which it is defined will execute; this determines which aggregations happen automatically (bottom-up). It is assumed that only abstract beliefs

can be in memory focus, since their formation is unconscious and a general characteristic of the functioning of memory. Note that only a limited number of basic aggregations exist, due to the limited number of arguments that can be abstracted from.

The second process type that the memory model supports is $complex\ aggregation$, and aggregates over retrievable beliefs to form a $complex\ belief$. A complex belief aggregates information from the level of the beliefs that were the input of the complex aggregation that formed it. These can be basic or abstract beliefs, as other complex beliefs. A complex aggregation reasons over arguments of beliefs taking a specific constraint into account. An example of a complex belief is: $complex\_belief(last, for(identity, contact2, hostile, free, free, free), with\_ranges(nr, nr, nr, any, any, any), has\_values(given, given, given, 4, vision, 0.9))$. It is formed by the following aggregation:

$$\begin{aligned} &agg(temporal\_source\_certainty\_aggregation(T, S, C),\\ &\quad highest\_free\_free\_free(\_, any, \_, any, \_, any, P, O, V),\\ &\quad is\_available\_at\_current\_time(belief(P, O, V, T, S, C))) \end{aligned}$$

Many complex aggregations exist, due to the wide variety of possible constraints. The constraint that is taken into account will be situation and task dependent. Therefore, whereas the process itself is automatic, the start of a formation of a complex belief is controlled.

For this the predicate $in\_task\_focus\_at(B, T)$ is used. Which beliefs are in task focus at a certain time should be determined top-down, based on the current goals of the agent. Complex beliefs that are in task focus are either retrieved from memory, or actively formed by the aggregation as which they are defined. In either case, their occurrence is always tailored to the current (part of the) task that is attended to. Basic beliefs in task focus are always attempted to be retrieved.

Activation History and Availability

All beliefs are at the time of their formation present in the part of the agent's memory referred to as its working memory (WM). In addition, it is possible that they are present in WM at a later time, after they have been retrieved from long term memory (LTM). When transferring beliefs from an agent's WM into its LTM, it is important to indicate when they were formed or retrieved in WM. For this the two-place predicate $active\_at(B, T)$ is used. For every $belief(P, O, V, T, S, C)$ that is found in the agent's belief base it holds that $active\_at(belief(P, O, V, T, S, C), T)$, with T the time it was formed, thus active in WM. This explicit reference to the time at which a belief is active in WM, is required to determine for later time points whether it is possible to retrieve it from LTM

and how easy this is. Those aspects depend on the availability of a belief, denoted by $has\_availability\_at(B, A, T)$.

In the literature many theories can be found on the factors that influence the chance that a belief is retrieved from LTM, and also a few computational models. It was decided to base the beliefs' availability determination on the one embedded in the cognitive theory and architecture ACT-R (Anderson et al., 2004). In ACT-R declarative knowledge is presented by chunks, whose activation values determine their availability. The full equation that determines a chunk's activation takes into account several aspects, but for this paper only the base-level activation is taken into account. The base level activation B_i reflects the recency and frequency of use of the chunk, and is calculated by:

$$B_i = ln(\sum_{j=1}^{n} t_j^{-d}) + \beta_i$$

n: The number of presentations for chunk i.

t_j: The time since the j_{th} presentation.

d: The decay parameter. Standard this one is set at 0.5 (Anderson et al., 2004).

β_i: A constant offset.

In the memory model proposed here, chunks are replaced by beliefs. A belief B's presence in WM , i.e., each $active\_at(B, T)$, is interpreted as a 'presentation' of a chunk. The constant offset β_i varies between the belief types. For basic and complex beliefs it holds that β_i represents the belief's initial impression value, denoted by $has\_impression\_value(B, I)$. For abstract beliefs, β_i represents its deduced impression value: $has\_deduced\_impression\_value(B, I)$.

The latter is the summation of the impression value of the belief that it was abstracted of, with an additional amount that depends on its abstraction type. Currently, each free argument of an abstract belief adds 0.2 to the deduced impression value of that abstract belief, so for an abstract belief of type $temporal\_source\_certainty\_abstraction$, this amounts to 0.6. This ensures that in principle beliefs at a certain abstraction level, i.e., beliefs of which some details are forgotten, are more available than beliefs at a lower level of abstraction.

The impression values of beliefs can be influenced by various factors, e.g., by the emotional response it triggered, or by its importance for the agent's current task. For the current paper it was determined that the impression value of new basic beliefs is determined by its importance for the current task, expressed by $has\_importance\_for\_task\_at(B, I, T)$. The complex beliefs that are top-down determined to be required for the task, and are therefore in task focus, inherit the same importance value as the basic belief from which their determination stems. The model has the ability to determine such

an initial value based on the context of the task; currently all new basic beliefs receive the same impression value of 2.

Retrieval and Aggregation Costs

In ACT-R a chunk's activation value does not only determine which chunk will be retrieved, but also how long it will take. The proposed memory model does not reason about the durations of specific retrieval processes, instead it attaches 'cognitive costs' to reasoning or retrieval actions. Intuitively, the costs to retrieve a belief should depend on its availability, therefore it was decided that its retrieval costs are inversely proportional to its availability. In addition, a threshold is required that denotes the minimal availability a belief should have in order to be retrievable. This threshold is arbitrary; for mathematical reasons it was set at 1. The costs of a complex aggregation are determined to be equal to the costs of the retrieval of all relevant beliefs, i.e., beliefs that it takes into account for its formation. So the costs of forming $complex\_belief(last, for(identity, contact2, hostile, T, S, C), with\_ranges(nr,\ nr, nr, any, any, any), has\_values(given, given, given, X, Y, Z))$, is equal to the summation of the retrieval costs of all retrievable beliefs $belief(identity, contact2, hostile, T, S, C)$.

4.3.3 Implementation

The developed model was implemented in Prolog (Wielemaker, 2003): http://human-ambience.few.vu.nl/docs/IAT-MemoryModel.pl. In this section, the implementation choices are explained using fragments of the program.

A run of the agent starts by requesting the start of a specific scenario. This sets the current time at 0, fills the agent's LTM with certain beliefs relevant for that scenario, and starts the agent's execution cycle by calling the `scenario_loop` clause:

```
scenario_loop(N):-
    current_time(T1),
    retract(current_time(T1)),
    T2 is T1 + 1,
    assert(current_time(T2)),
    determine_availabilities_beliefs,
    determine_availabilities_abstract_beliefs,
    sense_and_form_beliefs(N),
    set_task_focus(N),
    retrieve_beliefs,
    deduce_complex_beliefs,
```

```
    reason_and_form_beliefs(N),
    set_memory_focus(N),
    deduce_abstract_beliefs,
    scenario_end(N).
```

The scenario loop starts by propagating the current time with 1 time step. Next, the availabilities of basic and complex beliefs are determined following the ACT-R formula. The determination of the availability of abstract beliefs varies slightly from that of basic beliefs, and is therefore called separately. Basic beliefs always have a maximum of one `impression_value` attached to them, which is determined at their formation time. However, abstract beliefs might have multiple `deduced_impression_value` 's attached to them, because various basic beliefs may be abstracted into the same abstract belief. Therefore, the constant offset β_i of an abstract belief is determined to be the average of all its `deduced_impression_value` 's.

Next, a predicate is called with a specific reference to the scenario N that is executed; `sense_and_form_beliefs(N)` . Its clauses constitute the agent's working memory input stemming from the agent's interaction with the outside world. They define for each time step of a specific scenario which beliefs enter memory, together with their costs (fixed at 1) and impression value (fixed at 2).

After this, another scenario-specific predicate is called; `set_task_focus(N)` . Its clauses define for each time step of a scenario which beliefs are required from the task's perspective, and with which importance, see Section 4.3.2. The `in_task_focus_at` predicates embed belief templates of which an instantiation has to become active in . In case of a basic belief template this is attempted by a retrieval action. In case of a complex belief template it is first attempted to retrieve it, and when this fails, it is actively formed by executing the aggregation as which it is defined.

Because complex beliefs have a temporary nature their retrieval, instead of formation, allows for biases. Therefore, it is desired that this retrieval can be influenced, e.g., by stress. To enable that, the retrieval process of complex beliefs takes a specific, tunable, threshold into account. In the current scenarios this threshold is fixed at 100, so complex beliefs can never be retrieved and are always newly formed.

The `retrieve_beliefs` clauses try to retrieve instances from the agent's LTM for all the beliefs that are currently in task focus. For each of them it executes:

```
determine_retrieval_of_belief(BT):-
    current_time(N),
    retrieval_action(BT, B, RB),
    has_availability_at(B, A, N),
    assert(active_at(B, N)),
```

```
assert(active_at(RB, N)),
K is 1 / A,
assert(costs_to_retrieve_at(B, K, N)),
assert(costs_to_ perform_retrieval_at(RB, K, N)).
```

The `retrieval_action(BT, B, RB)` determines which belief `B` from LTM is retrieved based on the belief template `BT` that is in focus. Retrieved belief `RB` denotes the instantiation of `BT` with the values of belief `B` . The retrieval action retrieves that belief that is coherent to the requested belief template and has the highest availability value of all the coherent beliefs. Next, both `B` and `RB` are asserted to the agent's memory. The first because it is again active in WM, the second as a short cut for the reasoning process that will operate on it later. Then, the availability of belief `B` is queried and used to determine its retrieval costs, and thus of the formation of retrieved belief `RB` , which is the inverse of its availability.

An important feature or the memory model is the way in which the retrieval_action operates when a basic belief is the object of retrieval, e.g., `BT` is `belief(new_detected,contact2,_,free,_,_)` . This template can be filled by a basic belief, but also by an abstract belief, as long as the arguments that are designated as `free` in the template are among those where the latter abstracts from. This feature implements that, given that abstract beliefs have multiple instances and are as such more available that the specific beliefs they were abstracted from, abstract beliefs might still be retrievable while the detailed beliefs may not.

The execution cycle of the agent continues with the `deduce_complex_beliefs` predicate. Its clauses execute the aggregations as which the complex beliefs are defined that are in task focus, but could not be retrieved. For the details concerning the execution of these aggregations, see (Heuvelink et al., 2008b). Note that when an aggregation requires a complex belief as input, it is first attempted to retrieve it. When that fails, the required complex belief is formed by execution the aggregation at which it is defined. The result, so the complex belief that was formed as an intermediate step to the required complex belief which is in task focus, is also memorized. The costs of the formation of a complex belief by an aggregation are equal to the sum of all the basic beliefs that were retrieved for it, as explained in Section 4.3.2.

After all the beliefs that are required for reasoning have been retrieved or formed by aggregation in WM, the `reason_and_form_beliefs(N)` predicate is called. Its clauses specify for each time step of a scenario the specific reasoning rules that execute. These rules operate on the beliefs in WM and enter one or more new beliefs into memory, together with the costs of their generation. The latter is the summation of the costs to retrieve and/or aggregate the beliefs that functioned as the rule's input.

At the end of the scenario loop the memory's intrinsic feature of performing basic aggregations to form abstract beliefs is executed. The basic aggregations that happen are defined by the predicate `in_memory_focus_at` , and these are set by calling the scenario-specific predicate `set_memory_focus(N)` . In the current model it is assumed that at each time step all possible basic aggregations of the T, S, and C arguments of beliefs are in memory focus. This entails that from a single basic belief seven abstracted beliefs of different types are formed. The number of abstract beliefs that can be formed from an abstract belief depends on the abstraction level of the latter.

When `deduce_abstract_beliefs` is called, the basic aggregations that are in memory focus execute for each of the basic and abstract beliefs that are in WM. The process of abstracting basic beliefs into abstract beliefs that, when formed on the basis of multiple basic beliefs, are more and therefore longer available, resembles the transfer of episodic to semantic memory.

The last predicate of the scenario loop, `scenario_end(N)` , ensures that as long as the `current_time` is not equal to the defined end time of scenario N, `scenario_loop(N)` keeps being called.

4.3.4 Results

This section presents a scenario from the domain of (Heuvelink and Both, 2007) to demonstrate the results of the various processes within the memory model. The results here are obtained from the Prolog program. The table below shows for each time point 1) the **beliefs** that are newly formed based on input from the environment, and 2) the *beliefs* that the agent wants to form by reasoning.

How the beliefs are formed by reasoning depends on the reasoning rules that execute. This paper does not focus on the way in which it is determined which rules may execute, or whether rational or biased rules are selected. For a model that does cover the latter aspects, and can be used in conjunction to the memory model, see (Heuvelink and Treur, 2008). For the current research it is assumed that the selection of reasoning rules is simply done, and that these can either be rational or biased reasoning rules.

To demonstrate that the memory model can support rational and biased reasoning, two scenario versions are developed, named *rational* and *biased*. In the rational version the rational reasoning rules always require complex beliefs are input, which thus become in task focus. Given that the retrieval threshold of complex beliefs is set at a 100, these beliefs are always newly formed. This ensures that always the latest, most correct information is used for reasoning and therefore, this scenario performs rational reasoning.

The total costs of the *rational* scenario mount to 22.0445, which sums the costs to form the basic beliefs by sensing (set at 1: so 7 in total) and to form them by reasoning.

time	**Sensed** and *Required* Beliefs
1	**belief(position, self, [0,0], 1, gps, 1.0)**
2	**belief(position, contact1, [4, 3], 2, radar, 0.9)**
3	*belief(distance, contact1, _, _, _, _)*
4	**belief(position, contact2, [3, -1], 4, radar, 0.9)**
5	*belief(distance, contact2, _, _, _, _)*
6	*belief(within_weapon_range, contact2, _, _, _, _)*
7	**belief(position, contact1, [4, 2], 7, radar, 0.9)**
8	**belief(position, self, [0,1], 8, gps, 1.0)** *belief(within_weapon_range, contact1, _, _, _, _)*
9	*belief(distance, contact1, _, _, _, _)*
10	*belief(behavior, contact1, _, _, _, _)*
11	**belief(position, contact2, [2, -2], 11, radar, 0.9)**
12	*belief(distance, contact2, _, _, _, _)* **belief(behavior, contact1, receding, 12, officer, 0.6)**
13	*belief(behavior, contact2, _, _, _, _)*
14	*belief(threat, contact1, _, _, _, _)*
15	*belief(threat, contact2, _, _, _, _)*
16	...

The cost to form a belief by reasoning depends on the costs to retrieve the input that is required for the rule that forms it.

The screenshot below shows the costs to form for times 2-5 beliefs concerning contact1's en contact2's **position** and *distance*. Notice that the formation of the *distance* belief is cheaper for contact2 than for contact1. This is the case because of the formation of one of the required beliefs to determine this; the most certain belief about the ships own position taken into account the time passed since the formation of the basic beliefs about this. For this, all these basic beliefs are retrieved. Therefore, when later the same belief is required for determining the distance to contact2, its formation is cheaper. This is because the basic beliefs it is based on now have a higher availability value, and are thus easier to retrieve.

```
2 ?- costs_to_generate(B,K).
B = belief(position, contact1, [4, 3], 2, radar, 0.9),
K = 1 ;
B = belief(distance, contact1, 2.64575, 3, distance_from_
position_determination, 0.75),
K = 0.926153 ;
B = belief(position, contact2, [3, -1], 4, radar, 0.9),
K = 1 ;
B = belief(distance, contact2, 2.0, 5, distance_from_posi
tion_determination, 0.55),
K = 0.894508 ;
```

The rational reasoning that happens also ensures that when at time 14 the threat of contact1 has to be determined, this is based on the information that that contact is approaching, even though at time 12 an officer said the contact is receding. This is because the agent has deduced at time 10 that it is approaching, based on its beliefs about the contact's distances. Because these beliefs are based on beliefs about its positions, which are retrieved from radar and therefore have a high certainty, the belief about the contact approaching also receives a high certainty. The belief stemming from the officer on the other hand, received a lower certainty. Therefore, when at time 14 a complex aggregation reasons about which value of the belief about the behavior of contact it is most certain, it is correctly deduced that the contact is approaching, and consequently that it is threatening.

The reasoning rules in the *biased* scenario version do often not require complex beliefs are input, but simple basic beliefs, whose template thus become in task focus. As explained in Sections 4.3.2 and 4.3.3, basic beliefs in task focus are attempted to be retrieved from memory. In this version no conscious complex beliefs are formed to reason about a contact's distance. Instead of aggregating the last radar belief about a contact's position, it simply attempts to retrieve any belief about its position. Often this will be the latest, since this is likely to be the most available. However, this does not need to be, e.g., in the case an earlier belief has been retrieved more often.

Notice that biased rules do sometimes need a complex belief as input, e.g., for the rules that determine a contact's behavior. For that it uses the last, as well as second last belief about its distance. Such beliefs could never be retrieved by a basic belief template, since that only enables the retrieval of the belief with the highest availability and not also the second highest one. This problem was exactly the reason to implement the beliefs are they are; time-stamped (Heuvelink and Both, 2007). However, the aggregation that executes to determine the last and second-last belief in the biased scenario takes abstract beliefs as input, compared to basic beliefs in the rational scenario.

The total costs of the biased scenario mount to 20.1549, which sums the costs to form all the basic beliefs. The costs to form the beliefs at time 3 and 5 about contact1's and contact2's distances are 0.818635 and 0.757405 respectively. This is a bit cheaper than by rational reasoning. However, these costs will diverge much further when multiple basic beliefs exist concerning their, as well as the own position, which are required as input for deducing the distance. Multiple beliefs on the one hand decrease the costs of the retrieval of an abstract belief about them, which due to the multiple presentations has a higher availability. On the other hand they increase the costs of retrieval of the latest radar contact about it, since now more beliefs are retrievable and thus considered, which add to the cost.

As result of the biased reasoning that happens in the second version of the scenario, it is at time 14 simply retrieved what the behavior of contact1 is. Since the belief stemming from an uncertain source is the most available one due to its recentness, it is in this scenario falsely deduced that contact1 is not threatening.

4.3.5 Discussion and Conclusion

The memory model described in this paper is specifically designed to allow for both rational and biased reasoning, and can be used to form any type of aggregation. In these aspects it differs from most related work on memory models.

Memory storage is an element in all existing (cognitive) agent architectures. In a recent review study on computer-based human behavior representations (Morrison, 2003) it was generalized that 'all the (human behavior) models can represent either short term memory (STM) or long term memory (LTM)'. However, the ways in which these memories function differ greatly. For example, ACT-R's STM is formed by a retrieval buffer that can hold one chunk, which it retrieves using an activation function from its declarative memory module (LTM) (Anderson et al., 2004), while Soar's STM is formed by its working memory that is not limited in the number of elements it can hold (Laird et al., 1987). Although those architectures are designed to bring about a specific behavior and are not built to handle the abstractions and aggregations as described in this paper, it might be possible to map the specific belief construct to the memory construct of an architecture, provided that the form of the latter has a certain degree of freedom (Muller et al., 2008). For further mapping of the proposed memory model, the control mechanism for the formation of aggregations needs to be incorporated in the architecture, as well as the retrieval mechanism of beliefs from LTM into WM.

In addition to the general memory models, there exist specific 'episodic memory' models. In (Hintzmann, 1986) a simulation model of episodic memory is developed, which is used for the learning of concepts. Similar to the approach presented, it is assumed that abstracted knowledge is derived from a pool of episodic traces. The model exhibits basic findings from the schema-abstraction literature, such as differential forgetting of prototypes and old instances. But it does not allow for arbitrary aggregations, controlled by the agent's task.

As already mentioned in the introduction, Nuxoll and Laird (2004) also provide a model of episodic memory, which has been implemented in Soar (Laird et al., 1987). The model is based on the framework that they present, in which the most important design choices for each of the phases of episodic learning are described. Their framework allows for many possible events that could trigger automatic formation of episodes, and also for decay and removal of elements of the episode. However, in their model the

decay has not been implemented yet. Also, there is no abstraction or aggregation in their implementation: all details of an episode are always retrieved.

There are a number of considerations about the implementation of the model presented in this paper that are worth discussing. First, currently all the beliefs that are used to form an aggregation are retrieved into WM. This is done because it is desired that their availability is increased, i.e., they become more easily retrievable afterwards. Although the latter seems a natural choice, it might be more valid to implement a semi-retrieval, which increases their availability without actual retrieving them.

Secondly, the costs of forming a new basic belief by reasoning are currently equal to the costs of the retrieval and aggregation of its inputs. This means that the costs of the reasoning process itself are ignored. It is an implementation decision to take those costs also into account. Related to this, the costs of basic aggregations are also zero in the current implementation, and all basic aggregations happen automatically. Another choice would be to assign different costs to different aggregations. This would allow controlling the formation of basic aggregations by certain factors, e.g., stress: if the stress factor is high, only cheap aggregations are formed. This would implement another type of bias.

In the current stage of the work, the control of the formation and retrieval is parameterized, but the values for the parameters are not yet automatically set. An aim of future work is to extend the current model with a mechanism to automatically derive the values of the control parameters from task and situation aspects.

In addition, stress might be included to allow for agents that exhibit even more human-like reasoning. There are several ways in which a stress parameter might influence the process: first, it can influence which complex beliefs are formed; second, it can influence the reasoning rules that are executed and thus the beliefs that are set in task focus; finally, it could even determine which basic aggregations happen.

Chapter 5

Control Component

5.1 Introduction

In the previous two chapters we introduced a belief component and accompanying memory component that support the modeling of rational and biased behavior in military situational assessments tasks. Whether the agent would show rational or biased behavior was left unspecified. In this chapter we investigate how the control of an agent's reasoning process can be modeled in such a way that it dynamically determines, depending on the circumstances, whether the agent behaves rational or biased. In addition, we investigate information acquisition mechanisms that control whether the agent will attempt to retrieve required information from memory, or sense it in the world.

In Section 2.2.2 we made a distinction between 1) biases that operate *within* a process and influence *how* it executes, e.g., how in belief formation the trust in a source influences the believability of its information, and 2) biases that operate *between* processes and influence *which* executes, e.g., which source is checked. So far we modeled biases that take place *within* specific cognitive processes. In this chapter we model the reasoning control and information acquisition control of an agent in such a way that they determine which process executes. The control models thus operate *between* processes and can bias behavior by selecting a non-optimal process.

We start this section with discussing various aspects of control methods. Next, we present a quick overview of existing control methods, followed by an account of the approaches we selected for our studies.

5.1.1 Aspects of Control

A large variety of agent control methods have been proposed in the literature. In table 5.1 we specify four dimensions on which control methods can be classified. The dimensions denote properties at different levels, which means that their combination defines a specific method, and that all combinations can be found.

Centralized vs. Decentralized
Centralized agent control embeds one central information processing system in an agent. This information processor operates on the information from the agent's perceptual modules and generates outputs for its action modules through the fixed activation of various modules. *Decentralized* agent control does not equip the agent with one central information processor. Instead, the agent embeds multiple information processors, each of them capable of operating on (as subset of) the information received from the agent's perceptual modules, and of generating outputs for (as subset of) its action modules.
Reactive vs. Deliberative
Reactive agent control denotes that the behavior of the agent is steered by the input that its information processor(s) receives. A method that implements such behavior is called *data-driven reasoning*. *Deliberative* agent control denotes that the behavior of the agent is steered by certain desires or goals an agent has. A method that implements such behavior is called *goal-directed reasoning*, or *means-end analysis*.
Declarative vs. Procedural
Declarative agent control denotes that the agent can explicitly reason about which behavior it will display. For this it is required that the actions it can execute are declaratively specified. *Procedural* agent control denotes that the agent does not explicitly reason about which behavior it will display, but that this is predetermined by the agent modeler, or learned implicitly.
Type 1 vs. Type 2 vs. Type 3
These three types of control were recently distinguished by Gray (2007a). *Type 1* control denotes the execution loop of an agent: it specifies which output of one functional module is provided as input to another, which is usually fixed within an agent. *Type 2* control denotes the execution loop within a functional module. Gray (2007a) distinguishes three subtypes. First, there exist *non-cognitive* modules. The internal control of these modules that transforms the module's input to its output is not based on a cognitive theory, but is formed by, e.g., a mathematical formula or machine learning algorithm. Such a control predicts, but does not explain human behavior. Second and third there exist two types of *cognitive* modules which do embed a theory of internal control based on cognitive science. These two types differ in how they form their output: the internal control of the first type can determine on-line which output is appropriate given the module's input; for the second type this is predetermined off-line. *Type 3* control denotes the execution loop to perform a specific task. It embeds task-specific knowledge about which strategy or method to apply to accomplish a given task in a given task environment. This execution loop is not independent of the Type 1 and Type 2 controls embedded in the agent: the Type 3 control should fit within the constraints delivered by them.

Table 5.1: Aspects of Control Methods

5.1.2 Existing Methods for Modeling Control

We start this section with control methods developed by Artificial Intelligence. Next, we shortly describe some human control characteristics, and introduce various control methods from Cognitive Science. Last, we discuss control within integrated architectures.

Existing Methods for Modeling Control within AI

In traditional AI systems, e.g., the General Problem Solver (GPS) (Newell and Simon, 1963; Ernst and Newell, 1969), control is in general *centralized* and implementing a *deliberative*, *Type 2* reasoning method. In GPS this reasoning method was *procedural*; the means-end analysis that it implemented for finding operations that could be done on objects to bring the current state closer to the goal state, received a priori knowledge about which operations would aid for certain specific differences between the current and the goal state. As such this module is, according to Gray (2007a), a cognitive module for which it is determined off-line what the appropriate output is given the input.

The early AI system MYCIN (Shortliffe, 1976) also utilized *centralized* control implementing *deliberative*, *procedural*, *Type 2* reasoning. MYCIN was developed as expert system to support physicians in identifying bacteria's causing infections and by suggesting treatments. It did this by asking questions about the situation. Later on, NEOMYCIN was developed incorporating a *declarative* reasoning method, enabling the reasoning over which questions to ask to the physician to diagnose the bacteria. By implementing this declarative reasoning method, the inability to reason explicitly over the possible actions (questions to ask) as found in procedural control systems was overcome.

STRIPS (Fikes and Nilsson, 1971), developed for automated planning, is another classical AI system implementing *centralized* control, utilizing a *deliberative* (means-end analysis), *Type 2* reasoning method, but in a *declarative* way. This entails that it is declaratively specified which effects the performance of an action, implemented as an application of an operator, in a certain state has. This enables the explicit reasoning over which operator to select to decrease the gap between the current and goal state.

In reaction to this *centralized*, *deliberative* view of control Brooks (1986, 1991) introduced his *subsumption architecture*. Instead of one central processor that operates on input from an agent's sensors to generate output for its actuators, Brooks assumes multiple modules. Each module forms a behavior-producing system, and directly connects perception to action. So, Brook's subsumption architecture composes intelligent behavior into many 'simple' behavior modules, and incorporates *decentralized* control. In addition, it implements *reactive*, *procedural*, *Type 2* reasoning methods: the behavior modules connect limited, task-specific perception directly to the actions that require it.

Around the same time of Brooks's subsumption architecture blackboard systems emerged (see, e.g. Hayes-Roth, 1985; Engelmore and Morgan, 1988). Blackboard systems embed a central (passive) unit, the blackboard, whose content can be inspected and updated by the modules of the system. The blackboard serves as a common knowledge base through which the various modules can communicate. The control in blackboard systems is *decentralized* as the system embeds multiple information processors that usually implement a *reactive* (to the content of the blackboard), *procedural*, *Type 2* reasoning method. Because all these components react independently to the content of the blackboard, some form of global control is required that determines which module may interact with the blackboard. Therefore, each blackboard system also incorporates a module that embeds a *declarative*, *Type 3* reasoning method that keeps the behavior on track of the task.

Systems that combine deliberative and reactive control are generally referred to as hybrid control systems. The main idea behind hybrid control architectures is that the reactive part of the agent, which takes precedence over the deliberative one, ensures robustness, fast response times, and adaptability. The deliberative part of the agent ensures that longer term goal-oriented issues can be handled (Nwana, 1996).

Maes (1991) was among the first to bring some of the strengths of the deliberative and the reactive control paradigms together. She implemented an agent as a set of competence modules, each with STRIPS-like pre- and post-conditions, that are linked in a network. An example of a link between two modules is the successor-link, i.e., the post-condition of one is the pre-condition of the other. Modules receive an activation level that represents the relevance of the module in the current situation, which is determined by a spreading activation process operating on the network. The higher the activation level of a module, the more likely it is that this module determines what the agent will do. Based on experience, the competence module network is developed and changed, e.g., by adding or deleting links.

In the last two decades many agent designers have embraced the Beliefs, Desires, and Intentions (BDI) framework for modeling human behavior (see Section 2.3.1). Agents based on the BDI framework embed a *centralized*, *deliberative* control, supported by the desires and intentions. Whether the *type 2* reasoning method it embeds is *declarative* or *procedural* differs on whether the intentions (plans) are dynamically formed, or are pre-established: most BDI agents embed a database of prefixed plans, and thus embed procedural control.

Another recent development in agent control methods is the development of coordination languages. The origin of coordination languages lies in theoretical informatics that is concerned with verification of agent behavior and evaluation of agent performance.

Coordination languages explicitly distinguish coordination from computation, i.e., they separate the control of the modules of a system from their internal functioning. This separation facilitates the replacement of modules, and enables the executing of modules whose internal functioning is unknown. Coordination languages have been developed that allow for the specification of centralized control (Bonsangue et al., 2000; Ciancarini, 1996) as well as decentralized control (Barbuceanu and Fox, 1996), referred to as control-driven and data-driven coordination languages respectively. Bosse et al. (2007) developed a coordination language that can be used to specify both types of control approaches. Coordination languages usually specify *Type 1* control.

View on Control from Cognitive Science

Humans seem to possess central control (consciousness) but also decentralized control, e.g., when touching a hot stove the hand is retracted before the mind becomes conscious of the heat. In addition, humans display a combination of deliberative and reactive control. On the one hand humans are capable to guide their attention, e.g., they can decide where to look at. On the other hand, this deliberative control is limited in the sense that there exist so-called 'enduring dispositions' which humans automatically attend to (reactive), e.g., flashing lights (Kahnemann, 1973). Moreover, humans seem to be able to declaratively decide where to look at, e.g., when trying to find differences between two pictures, or more procedural, e.g., while scanning a new person. Last, it is self-evident that humans possess Type 1, Type 2 and Type 3 controls.

Davis (2004) investigated differences in the nature and requirements of biological and synthetic minds in terms of control: control over what is sensed; control over how that is perceived; control over how those perceptions are processed; and control over how this epistemic flow leads to control over actions. Davis does not provide answers how these various control mechanisms should be modeled, but put forward the thesis that they can be modeled using a systematic control language based on affects and affordances. 'Affect' is a more general concept than emotion since it covers other things besides emotions, including moods, attitudes, desires, preferences, intentions, dislikes, etc. (Sloman et al., 2005). Previously we mentioned that many agents are nowadays based on the BDI framework and as such embed a control method based on desires (goals). Also we followed this approach when modeling the agents introduced in the previous chapters.

Castelfranchi and Paglieri (2007) explore the structural interdependency of beliefs and goals in cognitive agents, and provide a model of the interplay of beliefs and goals in deliberation, planning and action. They define goal processing as 'the cognitive transition that leads from a mere desire to a proper intention'. They consider a goal as transferring through several states, see Table 5.2: from active goal (after its generation caused by a

Goal Type	Process Stage	Supporting Beliefs	Beliefs sub-classes	+/-
	ACTIVATION	Motivating beliefs	Triggering beliefs	+
			Conditional beliefs	+
Active Goals *(= desires)*				
			Self-realization beliefs	-
	EVALUATION	Assessment beliefs	Satisfaction beliefs	-
			Impossibility beliefs	-
Pursuable Goals				
		Cost beliefs		-
	DELIBERATION	Incompatability beliefs		-
		Preference Beliefs	Value beliefs	+
			Urgency beliefs	+
Chosen Goals *(necessary for future-directed intentions)*				
		Precondition beliefs	Incompetence beliefs	-
	CHECKING		Lack of conditions beliefs	-
		Means-end beliefs		+
Executive Goals *(necessary for present-directed intentions)*				
	ACTION → Feedback and subsequent (1) belief revision and (2) plan diagnosis			

Table 5.2: Model of belief-based goal processing from Castelfranchi and Paglieri (2007)

motivating belief) through a goal that the agent judges pursuable (through assessment beliefs about that the goal does not yet exist, or is self-fulfilling or impossible) to a goal that the agent actually chooses (with the help of preference beliefs that might use costs and incompatibility beliefs) and might execute (given that there is no belief about the incompetence of the agent or lacking preconditions).

Castelfranchi and Paglieri (2007) also introduce a *constructive theory of intentions*. In their view a goal becomes an intention only after passing the above mentioned series of screening tests, in which specific beliefs act as filters. Moreover, an intention can be *future-directed* or *present-directed*. An intention is future-directed when it is a *chosen* goal but not yet realizable, i.e., because preconditions miss. It is present-directed when it stems from an *executive* goal, i.e., a chosen goal that is immediately realizable.

Although an intention requires a goal at a specific stage of processing, Castelfranchi and Paglieri do not say that such a goal is the corresponding intention. Instead, they state that when a chosen goal becomes an intention a crucial transformation occurs and it immediately becomes in effect of that choice a double-faced entity. This entity includes both a *target* (what the agent wanted to achieve in the first place) and a *vehicle* (the action or plan that will achieve it). They refer to the closely related distinction between *intention-that* and *intention-to* but do not consider them as two different types of

intentions (as is customary), but as the two necessary elements of any intention.

They spell it out as follows: whenever an agent has the intention of doing something intentionally, this requires both the intention-to perform that action (Int-Act) and the intention-that one of the expected results of the action will hold after execution (Int-End). Each of these 'intentions' can be analyzed in terms of goals at a given stage of processing, but it is only the combination of the two of them that captures the exact meaning of *intending*. Moreover, the agent must be aware of the means-end relationship between doing A and achieving p, i.e., he must have the belief that doing A is a *means* to bring it about that p. They state that these three conditions, i.e. Int-Act, Int-End, plus the belief on Int-Act being a means for Int-End, are not only necessary for intentional action, but also jointly sufficient. In the next section we will elaborate on how these ideas relate to the work presented in this chapter.

Existing Methods for Modeling Control within Integrated Architectures

The Type 1 control that integrated architectures incorporate is procedural of nature. We are not familiar with any architecture whose general execution cycle is specified declaratively so it can be varied. Also many of the Type 2 reasoning controls embedded in current integrated architectures are procedural. This is the case for Soar, which evolved from the GPS paradigm and embeds prefixed *preferences* for operators, but also for ACT-R, which embeds prefixed, or implicitly learned *utility* values of productions. An exception is CLARION, which supports declarative reasoning about what to do through its meta-cognitive subsystem.

A consequence of the procedural, so fixed control structures embedded in cognitive architectures is that the behavior of the agent cannot be varied on-line: it behaves, given a certain input, in a fixed way. This became a problem for the developers of ACT-R when they wanted to use it to model task learning behavior. For task learning it is required to model explorative behavior, but modeling explorative behavior is incompatible with embedding a fixed way to select actions based on the utility values of productions. They circumvented this problem by attaching a random noise-value to the utility of productions.

Another consequence of procedural control structures is that it is important for a modeler to select the 'right' architecture whose Type 1 and Type 2 control matches with the Type 3 control required for the task he or she wants to model. This is important because the Type 3 control should fit within the constraints delivered by these procedural, thus fixed controls. For example, because the Type 1 and Type 2 controls embedded in the ACT-R theory determine that a maximum of one production rule can fire at a time, this also holds for every Type 3 control model implemented in ACT-R.

5.1.3 Selecting an Approach

In this chapter we develop two control methods for cognitive agents, one Type 2 control for reasoning and one Type 3 control for information acquisition. These two methods allow for the modeling of more human-like behavior because they incorporate the finding that human behavior varies, not only based on external task aspects, but also on internal cognitive aspects such as personality, stress, and exhaustion. Such internal aspects are often not supported by integrated architectures, and caused us to develop these two control methods that enable cognitive agents to (dynamically) display varied behavior.

The control methods we develop are *centralized*: all the information is processed centrally after which a choice is made. We made this choice because it facilitates the supervision of which behavior will be displayed, which is required for training simulations. In addition, the control methods both incorporate a *deliberative* reasoning method. With this choice we subscribe to our previous choice to model agent behavior using mentalistic notions as beliefs and goals, which was applied in the agents developed in the previous chapter. The incorporation of goals enables the explicit reasoning over which actions to execute to reach that goal. In order to do this the controls incorporate a *declarative* reasoning method, providing the means to reason about the processes.

In the first paper of this chapter we model the control of an agent's *reasoning* process. In order to determine on-line which cognitive processing component should be executed, it is important to denote declaratively which components are possible. In addition, for modeling deliberative control, it is required to denote the types of beliefs a component has as output, as well as the belief types it requires as input, which is done using STRIPS-like pre- and post-conditions. For modeling varied behavior it is in addition required to attach a value to a component that denotes how desired it is that it executes. This value is not prefixed based on the current goal (as the preference labels of Soar, or utility values of ACT-R), but is determined on-line based on the internal state. For the specific details concerning this process, see Section 5.2.

Various aspects of this reasoning control model are compatible with the ideas of Castelfranchi and Paglieri (2007), whose model of goal processing based on a frame of supporting beliefs was introduced above, but not yet known to us when we started to work on the research presented in Section 5.2. Here we shortly discuss how the work of Castelfranchi and Paglieri relates to this work.

First, Castelfranchi and Paglieri (2007) make a distinction between an intention's target (Int-End) and vehicle (Int-Act). A similar distinction is made in the papers presented in this chapter: an agent's goal (in the form of a belief) can be considered an Int-End, because a goal represents the status of the world that is desired by the agent (a belief about specific information). The Int-Act can be considered, in the case of the reasoning control,

as the cognitive processing component that the agent desires to execute because of the chance that it brings the desired world status about. The belief that a certain component c can bring about the goal g is formed by a belief that $component\_has\_output(c, g)$, which is equal to the means-end beliefs of Table 5.2.

Second, the reasoning control framework embeds constructs that could be considered equivalents of the assessment beliefs of Table 5.2. In specific: the reasoning control only gives priority to components (Int-Act) if their attached goal (Int-End) is not believed to be already the case, to be impossible, or to become true without interference.

Third, in order to start planning for a specific goal (when the selected component is not executable so the goal is future-directed) it is determined that it should not be the case that a plan for that goal already exists (in which case $component\_for\_goal(c, g)$ exists).

Fourth, when cognitive processing components are selected to become active, it is a condition that an active component's outcome cannot be something whose opposite is currently desired to be reached by another active component with a higher priority. This is an implementation of the incompatibility beliefs of Table 5.2.

We appreciate the work of Castelfranchi and Paglieri, because they explicate the states a goal transfers through. All these steps are required for every goal-directed agent, but are often modeled implicitly. Castelfranchi and Paglieri make these steps explicit, by specifying the supporting roles of beliefs in the phases of a goal. This has several benefits: first, it can guide the designer of a goal-directed agent in which aspects (supporting beliefs) need to be specified. Second, these supporting beliefs can be used to explain why an agent does, or does not have a goal in a specific state. For example, for explaining why an agent does not adopt a desire as a pursuable goal, it is possible to refer to the existence of a 'self-realization' belief about it.

In the second paper of this chapter we model the control of *information acquisition* for a specific task. This control determines, in terms of Castelfranchi and Paglieri, which Int-Act, i.e., a memory retrieval or sense action, to couple to a chosen Int-End because of the chance that it results in a belief about the desired information. When we started this research we aimed at developing a generic, type 2 information acquisition component that could determine the best way to acquire required information; either by retrieving it from memory of from sensing it in the world. The idea was that this component would deliberate on the possible actions to acquire the required information and on their respective costs and benefits, and then make a (possibly biased) choice. Although we did develop this component, referred to as 'rational' strategy in Section 5.3, we discovered that it was unable to represent the observed behavior of people executing a simple task. Instead, we hypothesize that the participants followed a strategy that did take some of the task circumstances into account, but for a large part pre-determined their behavior.

Therefore, we also developed several task-specific (type 3) control strategies that on average better captured the behavior of humans, see Section 5.3.

5.1.4 Chapter Overview

This chapter embeds two papers. The first paper (Section 5.2) focuses on the development of a generic approach to control the emergence of decision-making biases based on external and internal factors. For this we implement 1) a centralized, deliberative, declarative, type 2 control for the agent's reasoning process, and 2) a method that dynamically determines the agent's cognitive exhaustion level. The reasoning control selects which cognitive processing components execute, and can control the emergence of biases by applying heuristics while performing the task. For determining whether to apply rational or biased components, the reasoning control takes into account the agent's current, dynamic cognitive exhaustion level, as well as the cognitive costs of the (possibly biased) cognitive processing components.

The second paper in this chapter (Section 5.3) focuses on the development of task-specific approaches to control the way in which required information is acquired. For this we implement centralized, deliberative, declarative, type 3 controls for information acquisition that determine whether required information is best gathered 1) externally by sensing, or 2) internally by retrieving it from memory. First, we develop a rational control strategy that weights the pros and cons of these two types of actions by taking into account aspects as the chance of success and costs of failure. In addition, we develop more fixed 'heuristic' control strategies. The task model in which these control strategies are embedded is based on the memory model developed in Section 4.3.

In this paper we do not only develop models of human behavior, but also investigate human behavior in a simple task. This experimental data serves as inspiration for the development of the 'heuristic' control strategies. Moreover, we use the experimental data to determine for each person the best fitting parameters of the task model, among others the followed control strategy. When the developed task model is able to mimic human behavior, we are a step closer to the modeling of cognitive agents that can display human-like behavior.

Research Paper

5.2 Controlling Biases in Demanding Tasks

Abstract

Many aspects affect the way humans perform tasks, among others somebody's personality and current exhaustion level. Under varying conditions the quality of the performance is known to vary as well, for example, due to biases that occur. This paper introduces a cognitive control model addressing these aspects. It has been formally specified, tested in simulations for various scenarios, and formally analyzed.

This section is published as:
Heuvelink, A., and Treur, J.* Controlling Biases in Demanding Tasks. In B. C. Love, K. McRae, and V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (CogSci 2008), Cognitive Science Society, p. 1392-1397. July 23-26 2008, Washington, District of Colombia.
* Authors are listed in alphabetic order and can be regarded as having made a comparable contribution.

5.2.1 Introduction

Humans show a great variety in how they perform tasks. This variability in task performance may affect the quality of the performance. It is well-known that stress, fatigue, or high task demands can deteriorate task performance (Sanders, 1983). At the basis of these findings lies the fact that humans have a limited amount of cognitive resources. When a task becomes more demanding, these resources might become insufficient. To deal with this, humans tend to perform a task by applying cheaper cognitive reasoning steps, such as heuristics. These shortcuts often work well and might even be regarded as adaptive given their ecological validity (Gigerenzer et al., 1999). However, when the outcome of such a reasoning step deviates in a structural way from the rational outcome, it is called a bias.

The challenge addressed in this paper is to design a computational model for task performance that controls the occurrence of biases based on internal and external factors as mentioned. Various applications may benefit from such a model of human-like task performance. For example, it can be used to design virtual characters that play a role in simulations in which human aspects are important, like in realistic training environments and social games. Furthermore, such a model may help a software agent to better understand human behavior in cooperative task performance, and thus aid decision support.

In the next section various findings on human task performance are described in more detail. After this, a control approach is introduced that can generate variable task behavior. Next, this approach to control is applied to the control of biases in task execution, which results in a formally specified cognitive agent model. This model is tested in a case study, in which the agent operates in several scenarios under a variety of internal and external aspects. Finally, the model is evaluated and the research discussed.

5.2.2 Human Task Performance

Individual differences in cognitive characteristics entail variety in human task performance. In general some humans are more gregarious, impulsive, distractable, and less patient than others (Shields, 1983). At the same time humans manage the limited resources they have in certain ways, see e.g., (Johnston and Heinz, 1978). The allocation of cognitive resources is claimed to be flexible and under own control

Kahnemann (1973) stresses the idea that humans have a limited amount of cognitive resources. He states that there is no exact fixed amount, but that it is influenced by the arousal level of a person: the higher the arousal level, the more resources can be made available, up to a certain point. From that point on an increase in arousal may not result in an increase of available resources. McBride et al. (2007) reaffirm this and point out that

humans are able to perform multiple tasks at once, as long as the total sum of processing demands does not exceed the available resources. When the total sum does exceed this level of available resources, task performance will decline (Posner and Boies, 1971).

A method humans apply to bring down the cognitive demands of a task is the use of heuristics. These rules of thumb often work well in certain types of situations. Characteristics of heuristics are, besides context dependency, their simplicity and speed. However, when they deliver incorrect or inaccurate results they are also referred to as biases, for an overview see (Wickens and Flach, 1988). Cognitive biases are known to arise especially under stress of overload conditions (Baron, 2000) and have an immediate impact on the quality of task performance.

Hancock and Warm (1989) acknowledge that demanding tasks over time do, through some kind of physiologically mediator, influence cognition. They forward the thesis that tasks themselves are the major sources of cognitive stress, which others support (e.g., Matthews and Desmond, 2002).

5.2.3 Model Setup and Control Approach

Our research focuses on designing a cognitive agent model that can mimic the variability in human task performance. Therefore, it possesses multiple cognitive processing components that can perform the same task. Moreover, these components vary in content, so the model can also mimic the variability in the quality of the task performance. Some components are rational and generate the output in a correct way, others represent typical biases and 'forget' to take certain factors into account. Components that perform biased processing require less processing resources than rational ones, but they may generate incorrect outputs.

Furthermore, the model possesses a control method to determine which of the cognitive processing components may become active to generate a required output, see Figure 5.1. On the top level, above the dashed line, the control processes are shown, distinguished from the component processing. Input for this control process is coordination information about the various components and their input-output connections. The output of this control level is control information on which components should become active. Each component has two input layers: one for coordination information (the upper square at the left side of the component), and one for data information (the lower square). Output is also generated at both levels, depicted by the squares at the right side of a component.

The cognitive agent model decides which component(s) may become active based on the current external, as well as internal states. A major constraint is that the required processing resources of the to-be-selected components have to lie within the available

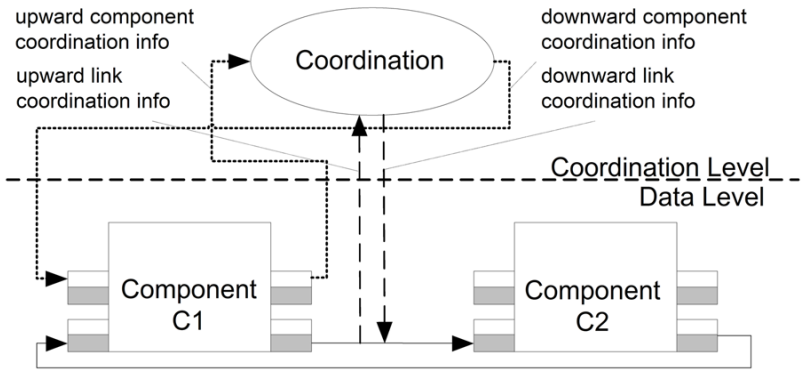


Figure 5.1: Control Approach

processing resources.

As discussed above, cognitive biases arise in human task performance under overload conditions. Since the model is about mimicking human (biased) task performance, the same principle should hold for an agent incorporating the model. The idea is that, when faced with high task demands, the agent will be motivated to operate on a high cognitive processing level. Over time, this will result in it becoming exhausted, which entails less available processing resources. The latter will affect the control decisions made. More specifically, when the agent becomes more exhausted, components with lower processing costs will be chosen, which usually implies a higher level of biases.

5.2.4 Formal Analysis

The cognitive agent model is expected to show certain behavioral properties as discussed above. Here such properties are identified and formalized, enabling automated verification. The first property relates task demand to biases:

HTDtoHB Higher task demand leads to higher biases.

This global property can be related to more local properties relating task demand to exhaustion, exhaustion to selection of less demanding components, and less demanding components to biases:

HTDtoHX Higher task demand leads to a higher exhaustion level.

HXtoLDC Higher exhaustion level leads to less demanding components.

LDCtoHB Less demanding components lead to higher biases.

The relationship between these behavioral properties is:

HTDtoHX & **HXtoLDC** & **LDCtoHB** $\Rightarrow$ **HTDtoHB**

For formalization of these properties a reified temporal predicate logical language was used; see e.g., (Galton, 2006). Expressions are built on atoms referring to state properties, time points, and traces. The properties can be formalized by comparing for one given trace the levels (of task and component demand, exhaustion level and biases) to certain bounds, or by comparing these levels in a relative manner in two traces. The following abbreviations are used:

aboveduring(γ, t, D, a(V), M) $\equiv$ $\forall$t1,V1 [t$\leq$ t1$\leq$ t+D & at(γ, t1, a(V1)) $\Rightarrow$ V1$\geq$ M

belowduring(γ, t, D, a(V), M) $\equiv$ $\forall$t1,V1 [t$\leq$ t1$\leq$ t+D & at(γ, t1, a(V1)) $\Rightarrow$ V1$\leq$ M

aboveleadstoabove(γ,D1, a(V), M1, E, D2, b(V), M2) $\equiv$

$\forall$t [aboveduring(γ, t, D1, a(V), M1) $\Rightarrow$ aboveduring(γ, t+E, D2, b(V), M2)]

aboveleadstobelow(γ,D1, a(V), M1, E, D2, b(V), M2) $\equiv$

$\forall$t [aboveduring(γ, t, D1, a(V), M1) $\Rightarrow$ belowduring(γ, t+E, D2, b(V), M2)]

belowleadstoabove(γ,D1, a(V), M1, E, D2, b(V), M2) $\equiv$

$\forall$t [belowduring(γ, t, D1, a(V), M1) $\Rightarrow$ aboveduring(γ, t+E, D2, b(V), M2)]

ishigherduring(γ1, γ2, t, D, a(V) $\equiv$

$\forall$t1, V1, V2 [t$\leq$ t1$\leq$ t+D & at(γ1, t, a(V1)) & at(γ2, t, a(V2)) $\Rightarrow$ V1 $\geq$ V2]

higherleadstohigher(γ1, γ2, D1, a(V), E, D2, b(V)) $\equiv$

$\forall$t [ishigherduring(γ1, γ2, t, D1, a(V)) $\Rightarrow$ ishigherduring(γ1, γ2, t+E, D2, b(V))]

higherleadstolower(γ1, γ2, D1, a(V), E, D2, b(V)) $\equiv$

$\forall$t [ishigherduring(γ1, γ2, t, D1, a(V)) $\Rightarrow$ ishigherduring(γ2, γ1, t+E, D2, b(V))]

Based on these, properties are formalized as follows:

HTDtoHBwithin(γ, D1, M1, E, D4, M4)

If in a trace γ for some time duration D1 the task demand is higher than M1, then after some delay E for some time duration D4 biases are higher than M4.

aboveleadstoabove(γ, D1, taskdemand(V), M1, E, D4, biaslevel(V), M4)

HTDtoHBbetween(γ1, γ2, D1, E, D4)

If in trace γ1 for some time duration D1 the task demand in γ1 is higher than the task demand in γ2, then after some time delay E, for some time duration D4 biases in trace γ1 are higher than biases in trace γ2.

higherleadstohigher(γ1, γ2, D1, taskdemand(V), E, D4, biaslevel(V))

HTDtoHXwithin(γ, D1, M1, E1, D2, M2)

If in a trace γ for some time duration D1 the task demand is higher than M1, then after some delay E1 for some time duration D2 the exhaustion level is higher than M2.

aboveleadstoabove(γ, D1, taskdemand(V), M1, E1, D2, exhaustionlevel(V), M2)

HTDtoHXbetween(γ1, γ2, D1, E1, D2)

If in trace γ1 for some time duration D1 the task demand in γ1 is higher than the task demand in γ2, then after some time delay E1, for some time duration D2 the exhaustion level in trace γ1 is higher than the exhaustion level in trace γ2.

higherleadstohigher(γ1, γ2, D1, taskdemand(V), E1, D2, exhaustionlevel(V))

HXtoLDCwithin(γ, D2, M2, E2, D3, M3)

If in a trace γ for some time duration D2 the exhaustion level is higher than M2, then after some delay E2 for some time duration D3 the demand of selected components is lower than M3.

aboveleadstobelow(γ,D2, exhaustionlevel(V), M2, E2, D3, componentdemand(V), M3)

HTDtoHXbetween(γ1, γ2, D2, E2, D3)

If in trace γ1 for some time duration D2 the exhaustion level in γ1 is higher than the exhaustion level in γ2, then after some time delay E2, for some time duration D3 the demand of selected components in γ1 trace is lower than the demand of selected components in trace γ2.

higherleadstolower(γ1, γ2, D2, exhaustionlevel(V), E2, D3, componentdemand(V))

LDCtoHBwithin(γ, D3, M3, E3, D4, M4)

If in a trace γ for some time duration D3 the demand of selected components is lower than M3, then after some delay E3 for some time duration D4 the biases are higher than M4.

belowleadstoabove(γ,D3, componentdemand(V), M3, E3, D4, biaslevel(V), M4)

LDCtoHBbetween(γ1, γ2, D3, E3, D4)

If in trace γ1 for some time duration D3 the demand of selected components in γ2 is lower than in γ1, then after some time delay E3, for some time duration D4 the biases in trace γ2 are higher than the biases in trace γ1.

higherleadstolower(γ1, γ2, D3, componentdemand(V), E3, D4, biaslevel(V))

Automated verification of these properties has been performed against generated simulation traces.

5.2.5 Dynamical System Models Used

In the next section the overall executable cognitive agent model is described. It includes some computational models in dynamical system style (based on difference / differential equations), which are introduced in this section.

The model is based on literature from cognitive science and human factors research. Hancock and Meshkati (1988) define mental workload as: 'The operator's evaluation of the attentional load margin (between their motivated capacity and the current task demands) while achieving adequate task performance in a mission-relevant context.' An elaboration on their figure illustrating this principle is shown in Figure 5.2.

Basic concepts used in our model are:

$x(t)$: the exhaustion level at t
mp: maximal cognitive processing level if no exhaustion exists
rp: relaxed cognitive processing level if no exhaustion exist
$td(t)$: the externally determined task demand at t
$ptd(t)$: the internally perceived task demand at t
$em(t)$: the effort motivation level at t
β: parameter determining source of effort motivation
$ap(t)$: the available processing level at t
$cp(t)$: the current processing effort at t

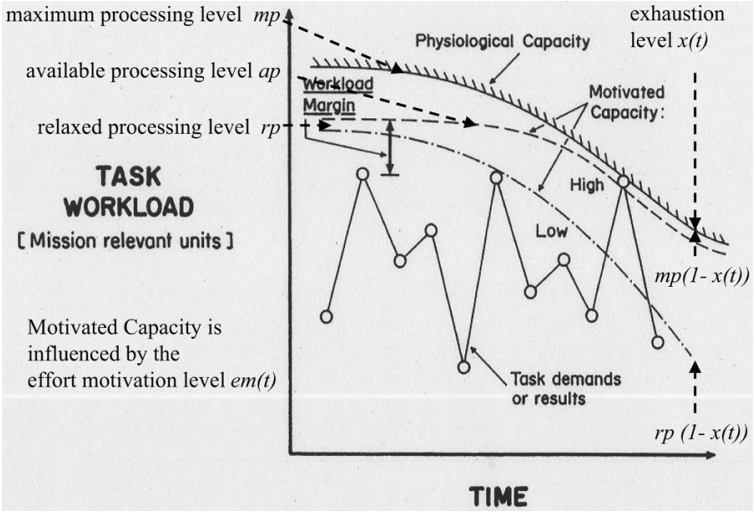


Figure 5.2: Cognitive processing levels over time

The exhaustion level $x(t)$ is assumed to be normalized between 0 (no exhaustion) and 1 (complete exhaustion). As exhaustion affects possible processing levels, the maximal cognitive processing level at some time point t is taken to be $mp(1 - x(t))$, and the relaxed cognitive processing level $rp(1 - x(t))$; this is illustrated in Figure 5.2. The incoming external task demand td is transferred to the internal perceived task demand ptd by dividing it by the current maximal processing level ($mp(1 - x(t))$). When the result is above 1, it is set to 1 which ensures that the perceived task demand lies between 0 and 1. The perceived task demand and exhaustion level determine based on a personality characteristic parameter β what the current effort motivation level $em(t)$ is, with a value between 0 (no motivation) and 1 (totally motivated). This level in return determines the current available processing level $ap(t)$, which influence the maximal processing effort $cp(t)$. Below these processes are described in more detail.

Exhaustion

First, the model for the level of cognitive exhaustion $x(t)$ over time is introduced. The exhaustion for a next time point depends on the current processing effort, but also on the current exhaustion, which is built up in the past. The assumption is that exhaustion increases proportionally to the amount by which the current cognitive processing effort $cp(t)$ exceeds the level indicated by $rp(1 - x(t))$. When the current processing effort is lower than this value, exhaustion decreases proportionally, until 0 is reached. Furthermore, the factor γ is used to fine-tune the model.

$$\Delta x = \gamma \frac{cp(t) - rp(1 - x(t))}{mp} \Delta t$$
$$x(t + \Delta t) = x(t) + \Delta x \quad \text{if} \quad x(t) + \Delta x > 0$$
$$= 0 \qquad\qquad \text{else}$$

Effort Motivation

At time t the cognitive effort motivation level influences the processing level at which the agent maximally operates. A personality characteristic parameter β is introduced that indicates in how far the motivation for effort is externally driven through the perceived task demand (indicated by $\beta = 1$), or internally driven by the exhaustion level (indicated by $\beta = 0$). The effort motivation $em(t)$ is determined as follows.

$$em(t) = \beta \times ptd(t) + (1 - \beta) \times (1 - x(t))$$

Available Processing Level

Given the motivation indicator the cognitive processing level made available $ap(t)$ is determined as follows. If the motivation is 1, the maximal possible processing level $mp(1 - x(t))$ will be the processing level made available. If the motivation is 0, the available processing level is the relaxed processing level $rp(1 - x(t))$, which is always proportional to $mp(1 - x(t))$. The general model for the processing level made available is:

$$ap(t) = (em(t) \times mp + (1 - em(t)) \times rp) \times (1 - x(t))$$

When $cp(t) = ap(t)$ is taken (i.e., the processing level made available is fully used), the three models for $x(t)$, $em(t)$ and $ap(t)$ above can be combined to obtain a single (but complex) difference or differential equation model for $x(t)$, given the chosen values $cp(t)$ for the current processing effort over time.

5.2.6 Overall Cognitive Agent Model

This section describes the overall design of the cognitive agent model, incorporating the dynamical models of the previous section. To evaluate whether the model indeed dynamically adjusts its task performance in a way similar to humans, it has been designed in a formal, executable format. The model includes various cognitive components and control knowledge about them. In addition, it is able to observe the world, form goals, and execute actions. The model's execution cycle is as follows:

Determine Observations: The agent observes the world and forms beliefs about what it sees.

Determine Goals: Based on beliefs, it forms goals with priorities.

Determine Task Demand: Based on the formed goals, their priorities, and the cognitive processing level that is required to reach them in the optimal way, the task demand is determined.

Determine Perceived Task Demand: The perceived task demand is deduced from the real task demand (see section above).

Determine Effort Motivation Level: see section above.

Determine Available Processing Level: see section above.

After these processes the agent starts the selection process of the cognitive processing components to be executed.

Determine Executability of Components: First, it determines which components are eligible for execution, i.e. that they can actually produce outputs when selected. For this it checks for each component whether all the input it requires is available.

Determine Relevance of Components for Goals: Next, it determines which components are relevant for which goal:

$\forall g\ \forall c\ \forall k\ \forall cr\ \forall kr\ \forall x\ [$

If $goal(g)\ \wedge\ component\_has\_output(c, g)\ \wedge\ component\_requires\_processing\_level(c, k)\ \wedge$
$component\_has\_output(cr, g)\ \wedge\ component\_requires\_processing\_level(cr, kr)\ \wedge$
$\neg\exists co\ \exists ko\ [\ component\_has\_output(co, g)\ \wedge\ component\_requires\_processing\_level(co, ko)$
$\wedge\ ko > kr\ ]\ \wedge$
$component\_has\_executability(c, 1)\ \wedge\ exhaustion - level(x)\ \wedge\ b = 1 - |(1 - k\ /\ kr) - x|$
Then $component\_has\_relevance\_for\_goal(c, g, b)\ ]$

This process entails that the relevance of a component c for a certain goal g that it has as its output, depends on the current exhaustion level x and the existence of a most expensive component cr that has goal g as its output. The rational behind this process is that the most expensive component is the best (most rational) component to reach *g*, and is preferred when there is no exhaustion (receives a relevancy of $1 - |(1 - k\ /\ kr) - 0| = 1$). However, the more exhausted the agent is, the more it prefers the cheaper components over the expensive ones. For example, when x is 0.3, cr only receives a relevancy of 0.7, while c, given it requires a lower processing level, e.g. 4 instead of 6, receives a relevancy of $1 - |(1 - 4\ /\ 6) - 0.3| \approx 0.97$. If a component does not have a certain goal as one of its outputs, its relevance for that goal is 0.

Determine Priority of Components: The priorities of the components for the various goals are determined by multiplying their relevancy for a goal with the priority of that respective goal.

Determine Components to be Activated: This is done by considering all possible groups of components for which it holds that 1) they have a priority greater than 0; 2) they are not relevant for the same goal; 3) their output does not make the goal of the other irrelevant. Furthermore, 4) their combined required processing level is below, or equal to, the available processing level. The components that are selected for execution are the members of the group with the highest total priority, which is formed by the sum of the priorities of the components.

Determine Activated Components: The components selected for activation are executed.

Determine Current Processing Effort: The current processing effort that the executing components deliver is determined.

Update Exhaustion Level: Given this current processing effort, the exhaustion level is updated, see the previous section.

As long as observations are made, the agent keeps controlling its process as indicated. When there is no task demand the agent relaxes, resulting in decreased exhaustion.

5.2.7 Simulation Experiments

To evaluate the cognitive agent model, simulation experiments were performed in LeadsTo (Bosse et al., 2007), which is especially developed to model executable temporal properties. For the evaluation a simple classification task was chosen. Although simplified, it is representative for the kinds of tasks future software agents might perform, e.g., in training simulations of military air-traffic-controllers. The task entails the correct classification at every execution cycle of the objects (none, one or two) then present in the world. The classification of an object entails assigning it to one of the bins present in the environment. Objects have two properties, namely a color (red, blue or green), and a shape (cube, cylinder or triangle). This results in nine possible objects that can be classified. Bins also have two properties: they can either fit red, blue, green, or all colors and cube, cylinder, triangle, or all shapes. For the current scenario's it is assumed that these 4 bins are present:

Bin 1: fits red cubes
Bin 2: fits blue objects
Bin 3: fits any colored triangles
Bin 4: fits all objects

The general goal of the task is to classify objects, but also to do this as precise as possible. The assignment of an object to a bin whose properties it exactly matches has the highest preference. Furthermore, partial classifications are desired above an assignment to the most general bin. So in the current scenario, the best classification of the red cube is assigning it to bin 1, followed by an assignment to bin 4. A blue triangle can be assigned to bin 2 just as well as to bin 3; bin 4 however is less desired.

To test the behavior of the cognitive agent model over time, four scenarios were developed. They all incorporate the same bins, but the objects present in the world over time differ. In the scenario named *1 object*, one object is present at every execution cycle. The similar principle holds for the scenario named *2 objects*. For scenarios named *low demand* and *high demand* the amount of objects varies, see Table 5.3 for an overview. Each scenario takes 9 time steps.

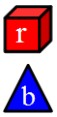
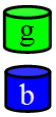
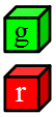
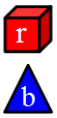
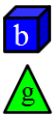
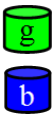
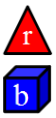
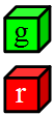
Time:	1	2	3	4	5	6	7	8	9
low demand									
high demand									

Table 5.3: Objects Present in World over Time

During the execution cycle of the model, the agent first observes the world and forms beliefs about the properties of the available objects and bins. Then it derives new goals from the top level goal $classify\_all\_objects$ as follows:

$\forall x\ \forall p$ [

If $goal(classify\_all\_objects)\ \wedge\ goal\_has\_priority(classify\_all\_objects, p)\ \wedge\ belief(object, x)$

Then $goal(belief(classification\_type\_of, x, total)) \wedge$

$goal\_has\_priority(belief(classification\_type\_of, x, total), p/3 \times 1.1) \wedge$

$goal\_satisfied\_when(belief(classification\_type\_of, x, total), belief(classified, x)) \wedge$

$goal(belief(classification\_type\_of, x, partly)) \wedge$

$goal\_has\_priority(belief(classification\_type\_of, x, partly), p/3 \times 1.0) \wedge$

$goal\_satisfied\_when(belief(classification\_type\_of, x, partly), belief(classified, x)) \wedge$

$goal(belief(classification\_type\_of, x, not)) \wedge$

$goal\_has\_priority(belief(classification\_type\_of, x, not), p/3 \times 0.9) \wedge$

$goal\_satisfied\_when(belief(classification\_type\_of, x, not), belief(classified, x))$]

So for every object the agent forms three classification goals, with varying priority. These priorities express the agent's preferences for the various types of classifications.

The task demand for the current task is determined by the combined task demand of the present objects. Objects entail task demand because they cause goals with priorities.

$$task\_demand = \sum (mp \times p \mid goal(g)\ \wedge\ goal\_has\_priority(g, p) \wedge maximum\_required\_processing\_level\_for\_goal(g, mp))$$

For the current task this entails that a single object delivers a total task demand of 5.06667. This results in a constant task demand for the scenarios *1 object* and *2 objects*: 5.06667 and 10.1333, respectively. For scenarios *low* and *high demand* the task demand varies, see Figure 5.3.

Above it was described how, based on the goals and the priority of components for these goals, the cognitive agent model determines which components execute. Besides the components themselves, it also uses control knowledge over these components, e.g.

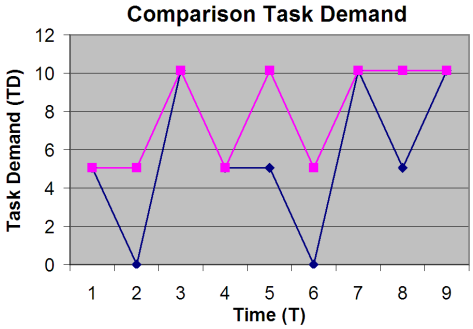


Figure 5.3: Task demand for scenario's *low demand*(blue diamonds) and *high demand* (pink squares)

about their inputs, outputs, and required processing level. The latter value is deduced from the number of required inputs of the component.

The following shows the process of a rational component in the form of an executable temporal rule:

$\forall x\ \forall p$ [
If $HoldsAt(belief(object, x), t)\ \wedge\ HoldsAt(belief(bin, y), t)\ \wedge$
$HoldsAt(belief(has\_shape, x, s), t)\ \wedge\ HoldsAt(belief(fits\_shape, y, s), t)\ \wedge$
$HoldsAt(belief(has\_color, x, c), t)\ \wedge\ HoldsAt(belief(fits\_color, y, c), t)$
Then $HoldsAt(belief(classified\_as, x, y), t+1)\ \wedge\ HoldsAt(belief(classified, x), t+1)\ \wedge$
$HoldsAt(belief(classification\_type\_of, x, total), t+1)$]

This component requires a processing level of 6 and has a bias level of 0. Besides rational components, biased ones are present with a different process but a same output, e.g.:

$\forall x\ \forall p$ [
If $HoldsAt(belief(object, x), t)\ \wedge\ HoldsAt(belief(bin, y), t)\ \wedge$
$HoldsAt(belief(has\_shape, x, s), t)\ \wedge\ HoldsAt(belief(fits\_shape, y, s), t)$
Then $HoldsAt(belief(classified\_as, x, y), t+1)\ \wedge\ HoldsAt(belief(classified, x), t+1)\ \wedge$
$HoldsAt(belief(classification\_type\_of, x, total), t+1)$]

This component also deduces a belief about a total classification, but forgets to take the colors of the object and the bin into account. The final result might be correct, but might also be incorrect. This second component requires a processing level of 4 and has a bias level of 4/6, because the most expensive processing requires a level of 6; see the previous section on the relevance of components.

Last, various parameters present in the model are assigned a fixed value to arrive at an executable version. For the current task the maximal processing level mp is set to 10 and the relaxed processing level rp to 7. It is assumed that the agent is not exhausted

at the beginning of the task. Furthermore, parameter γ, with which the granularity in exhaustion level over time can be tuned, is set to 0.3.

Each scenario was executed twice, once with personality value 0.7 (motivation primarily determined by external task demand) and once with value 0.3 (motivated primarily determined by internal exhaustion level).

Simulation Results

Scenario 1 Object

In this scenario the cognitive agent model classified each object in a perfect way for both personalities. Since there is a maximum of one object at each execution, the maximal possible current processing level (for the red cube classification) lies at 6. This is below the relaxed processing level, set at 7, and thus no exhaustion occurs.

Scenario 2 Objects

In this scenario the two objects ensure a constant high task demand of 10.1333. This results in a constant perceived task demand ptd of 1, which causes both agents to make more than their relaxed processing level available. Therefore the effort of the selected processing components often lies above the relaxed processing level $rp(1-x(t))$, causing the agent to become exhausted, which in turn influences the available processing level, see Figure 5.4.

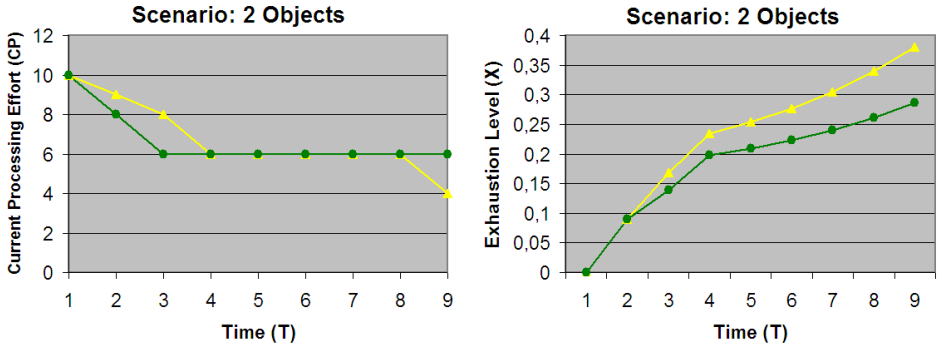


Figure 5.4: Current processing effort and exhaustion level for personality 0.7 (yellow triangles) and 0.3 (green dots)

The agent with personality value 0.7 will, given the ptd of 1, make more processing level available than the agent with personality value 0.3. This is beneficial at first; more available processing level entails that more demanding, so less biased, components can execute. However, due to this higher effort level this agent becomes exhausted quicker. This results in that it over time actually has less processing level available, which result

in the selection of cheaper and thus more biased components, see Figure 5.5.

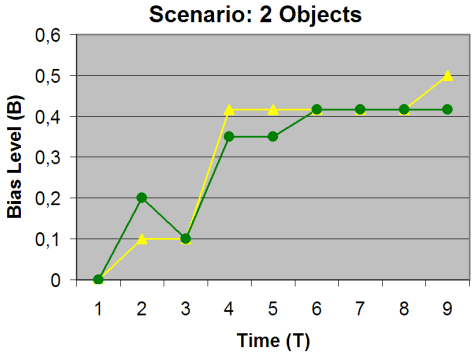


Figure 5.5: Bias level for personality value 0.7 (yellow triangles) and 0.3 (green dots) in the *2 objects* scenario

Scenarios low and high demand

In the scenarios *low* and *high demand* the numbers of objects that are available at each execution cycle vary, see Table 5.3. The variety in the task demand, see Figure 5.3, clearly determines the variety in current processing effort, see Figure 5.6. This in turn influences the exhaustion level and bias level, see Figure 5.7.

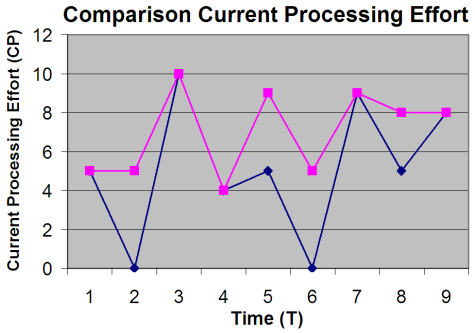


Figure 5.6: Current processing effort for the *low demand* (blue diamonds) and *high demand* (pink squares) scenarios

The increase in bias level has its impact on the quality of the task performance. Table 5.4 sums for personality value 0.7 the percentage of false classifications averaged over all objects present. As an example: this percentage is 75 percent when the agent blindly assigned an object to any bin, since it is correct for bin 4, which is one of four bins.

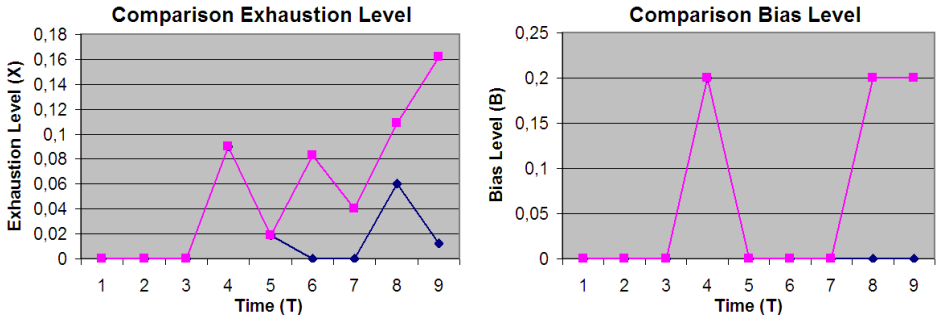


Figure 5.7: Exhaustion and bias level for the *low* (blue diamonds) and *high demand* (pink squares) scenarios

Scenario	1	2	3	4	5	6	7	8	9
1 object	0	0	0	0	0	0	0	0	0
2 objects	0	0	50	25	37.5	25	12.5	37.5	62.5
low demand	0	0	0	50	0	0	0	0	0
high demand	0	0	0	50	0	0	0	100	75

Table 5.4: Percentage of mistakes for personality value 0.7

5.2.8 Verification

Formalized properties, such as those presented earlier, have been automatically verified against a number of simulation traces, such as discussed above. As an example, property **HTDtoHBwithin** has been verified and shown to hold for all four traces for the following values for the duration and bound parameters: $D1 = 100$, $E = 100$, $D4 = 100$, $M1 = 8$, $M4 = 0.2$.

Notice that one execution cycle of the model takes a 100 time steps. Moreover, the property **HTDtoHBbetween** that compares two traces was also verified and shown to hold for the *low demand - high demand* scenario pair as well as the *1 object - 2 objects* scenario pair for the values: $D1 = 100$, $E = 100$, $D4 = 100$.

5.2.9 Discussion and Conclusion

This paper presented a cognitive agent model capable of dynamically adapting its behavior to the external, as well as its internal state. Related research with a similar goal focuses on integrating emotions, arousal, and motivation in cognitive systems, but no similar approach can be found. Closest to this work is the work of Ritter et al. (2007b) that implements various theories of stress and their effect on behavior (some considered biases). However, the implemented factors were local, fixed and no temporal aspect is in-

corporated. One theory they did not implement is that tasks themselves are stressors (the approach taken in this paper). About this they state 'we recognize that modeling tasks as stressors is an interesting and important next step in the effort to model the effects of stress.'

The dynamical cognitive agent model was tested for various task scenarios in simulations. A formal analysis of properties of the model has been performed, including automated verification of the identified properties against simulation traces, indeed showing the behavior as expected.

For a number of choices that were made for the case currently presented, also alternative choices could have been made, e.g., for the choice of parameters for the maximum en relaxed processing power in relation to the required processing level of components. It is expected that the values of the parameters depend on the application context. Based on the requirements of the behavior that the model should show, these can be adapted as to provide a best fit.

The model's main contribution is that it offers a mechanism to control the appearance of biases in a wide variety of tasks, but even stronger, on multiple levels of the task execution. The current paper solely addressed the controlling of biases appearing in cognitive components processing beliefs. The processes from observations to beliefs and from beliefs to goals were fixed. These processes may just as well be subject to biases under stress. A biased determination of priorities of goals can also have serious effects on task execution. In future work the control of these processes will be added to the current model.

Research Paper

5.3 Modeling Human Information Acquisition Strategies

Abstract

In this paper we focus on the development of a computational model that provides intelligent agents with a mechanism to decide on whether to acquire required information by retrieving it from memory or by interacting with the world. First, we present a task in which choices have to be made between acquiring information from memory or from the world. Two conditions are introduced with variable costs, and an experiment is performed to detect whether humans apply some kind of rational expected utility analysis to make this decision. Results indicate that humans do not, but instead adopt a simpler heuristic strategy. Next, we introduce a computational model that incorporates various heuristic task strategies, as well as rational ones. The human data is compared to the behavior of the model under various parameter settings. We were able to match the human actions with model actions for various task strategies, suggesting that humans differ in the task strategies they apply, and that our manner to deduce heuristic task strategies is feasible.

This section is an extended version of a paper published as:
Heuvelink, A., Klein, M. C. A., and Lambalgen, R. M. van.* Modeling Human Information Acquisition Strategies. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society* (CogSci 2009), Cognitive Science Society. July 30- August 1 2009, Amsterdam, the Netherlands.
* Authors are listed in alphabetic order and can be regarded as having made a comparable contribution.

5.3.1 Introduction

For the execution of almost all tasks knowledge is required. For example, making a phone call to a good friend requires - apart from procedural knowledge on how to operate a phone - explicit knowledge about the phone number. When preparing for the task, a human will make an (often implicit) choice between retrieving the required knowledge from memory, or looking it up. Intuitively, this choice is determined by the balance between the costs of looking up information on the one hand, and the costs of retrieval and the risk of mistakes on the other hand. In the phone call example the choice could be to retrieve the phone number from memory as a number of a good friend is probably easily retrievable, while the costs of looking up the required information are probably relatively high (finding the number in the address book), and the costs of mistakes are low (apologizing and calling again).

Selecting actions based on their expected costs and benefits is often described as rational decision making. However, it is well known that humans do not always follow a rational process, but often depend on heuristic approaches to solve a problem (Tversky and Kahneman, 1974; Gigerenzer et al., 1999). In addition, humans vary (between-subject) in the task-specific strategies they apply, but this choice is also influenced (within-subject) by the specific task circumstances (see, e.g., Beilock and Decaro, 2007; Byrne et al., 2008). For example, some people might prefer to always first try the phone number that they remember and only look it up in case of failure, even in cases in which a rational analysis would conclude that it is more efficient to look up the information.

The overall aim of our work is to build intelligent agents that exhibit human-like behavior. In order to do so, we would like to build a computational model that can decide on whether to acquire information by retrieving it from memory (information in-the-head) or by interacting with the world (information in-the-world).

In the first part of this paper, we describe an experiment with two cost conditions in which we analyzed the behavior of humans in a relative simple task. This task required the participants to choose between information in-the-head and information in-the-world. After elaborating on the task we discuss how rational expected utility analysis could be applied to it, i.e., what the possible task actions are, and the associated types of costs and benefits. Subsequently, the behavioral experiment and its results are presented.

In the second part of the paper, we try to align the results of the experiment with a developed task model that takes both the rational-choice approach and heuristic-based approaches into account. We first describe this task model and the possible heuristic task strategies that people could apply. Then, we describe the technical experiment by means of which we searched for values for the model's parameters that best fit the results of the behavioral experiment. Finally, the implications of the findings are discussed.

5.3.2 Task Description

The computer task we developed required participants to classify presented objects to specific bins. During the task, nine objects were presented in a sequence of thirty-six trials. The objects were composed of a color (red, blue or yellow) and a shape (square, circle or triangle). Each object belonged to a specific bin, numbered 1 to 9, but initially the participants did not know the correct combinations. The goal of the task was to press the number of the correct bin upon presentation of the object. On each trial participants had the option to press the number of a bin first ('choose'), or to press a button to get more information about the bins ('sense'). Participants could choose one of three buttons: button 'j' revealed the bins of objects with the same color as the presented object; button 'k' revealed the bins with the same shape; and button 'l' revealed the bin of the specific object. After the information was shown, participants had to select a bin. After a bin was chosen, the correct bin was revealed.

Participants started the task with 10 euro. Money was subtracted when either a button was chosen (Button Costs), or an error was made (Error Costs); see Table 5.5 for the two specific cost-settings used. In addition, for every 500 ms 0.01 euro was subtracted.

A typical trial started with presenting an object with below it nine empty boxes. Furthermore, the three buttons were shown and in the upper right corner the amount of money left.

When participants choose to sense color or shape, they had to wait for 1.0 seconds until the requested information was shown (Button Time). When participants choose to sense all, they had to wait for 1.5 seconds (Button Time). Meanwhile, time costs were still subtracted. When the waiting time had passed, the object was presented again with below it the nine bins, but this time the bins were revealed that matched the specific feature that was sensed: the three bins that matched the color of the object; the three bins that matched its shape; or the bin that matched the whole object.

When a bin was chosen (immediately, or after sensing), the object and the nine bins were presented again with the correct bin revealed. At the same time feedback was given on the choice of the participant.

Condition	Feature	Button Costs	Button Time = Extra Button Costs	Error Costs
1	Color	€ 0.10	1.0s = € 0.02	€ 0.10
1	Shape	€ 0.10	1.0s = € 0.02	€ 0.15
1	All	€ 0.15	1.5s = € 0.03	€ 0.20
2	Color	€ 0.06	1.0s = € 0.02	€ 0.12
2	Shape	€ 0.06	1.0s = € 0.02	€ 0.18
2	All	€ 0.09	1.5s = € 0.03	€ 0.24

Table 5.5: Costs of the two task conditions.

The combination of nine objects in thirty-six trials was determined previous to the experiment, to make sure that some objects would be often encountered so that over time it would be well known to which bin they belonged, while for others, less encountered, this could have been forgotten. See Table 5.6 for the number of specific objects presented over the trials.

Feature	3 x Red	2 x Blue	1 x Yellow
3 x Circle	RC: 9x	BC: 6x	YC: 3x
2 x Square	RS: 6x	BS: 4x	YS: 2x
1 x Triangle	RT: 3x	BT: 2x	YT: 1x

Table 5.6: Overview of objects presented.

Rational Expected Utility Analysis

The presented task requires interactive behavior: for its performance a mixture of elementary cognitive, perceptual, and motor operations are required. Gray and Boehm-Davis (2000) introduce interactive routines as the basis of interactive behavior. They envision interactive routines as dependency networks of low-level cognitive, perceptual, and motor operators that come together at a time span of about 1/3 to 3 seconds. Gray and Fu (2004) propose that at this time span, the human control system selects sequences of interactive routines that tend to minimize performance costs measured in time while achieving expected benefits.

For the presented task it is possible to rely to a smaller or larger degree on information in-the-world versus information in-the-head. In the first case more interaction with the world is required (button pressing), in the second case more intensive memory use (remembering the colors and shapes of the bins). Based on the specific task conditions it is expected that humans will adopt different interactive routines to minimize performance costs.

A rational strategy for performing the presented task would determine at each trial which of the four possible actions would be most optimal to execute: either directly choosing a bin, or first requesting which bins fit the color, shape, or both these aspects of the presented object. For this, a cost-benefit analysis of each action needs to be made.

For the presented task four types of costs exist: 1) the money it costs when a certain mistake is made, 2) the money it costs to press a button, 3) the time it costs to do so, and 4) the time it costs to retrieve beliefs from memory. It is possible to express all the various types of costs in money, because time costs money. It could be debated that in addition to these money and time costs another type of costs exist, namely the

cognitive and perceptual-motor effort involved in executing the actions. We do not separately distinguish these efforts but assume that time is a reasonable surrogate measure for them (Gray and Fu, 2004). By doing this we also decline the minimum memory hypothesis that suggests that humans are biased to conserve cognitive resources by favoring perceptual-motor resources (Wilson, 2002). Gray et al. (2006) make a convincing case that people do not favor strategies that minimize the use of memory, but those that minimize temporal cost-benefit tradeoffs.

To determine the expected utility of each of the possible actions, the expected costs for each of the four types of costs need to be determined. The money and time it costs to press one or none of the buttons depends on the task condition, but apart from that can be determined in a straightforward way. It is harder to determine the expected costs of 1) making an error and of 2) retrieving beliefs from memory.

For the first aspect the chance that one of the three possible errors is made (color false, shape false, all false) is important together with their respective, task condition dependent, penalties. The chance that a specific error is made depends on what is remembered. When it is possible to retrieve the correct bin for a specific object, the chance on any error is zero. However, when this is not possible the chance on a specific error depends on the chance of correctly retrieving knowledge concerning bins with the to-be-classified object's color or shape, but also on the chance that knowledge is retrieved that exclude specific bins from selection, increasing the chance the correct bin is picked.

The expected cost of retrieving beliefs from memory is equal to the time to do so or to the time to failure. These times, as well as the chance that knowledge can be retrieved in the first place, are important to know for calculating the expected utilities. Insight in these aspects can come from models of human memory. A well known model of memory retrieval is embedded in the cognitive theory ACT-R (Anderson et al., 2004). In ACT-R declarative knowledge is presented by chunks, whose activation values determine their chance and speed of retrieval, the latter according to this formula:

$$RT = Fe^{-A_i}$$

RT: The time to retrieve the chunk in seconds.

A_i: The activation of the chunk i which is being retrieved.

F: The latency factor parameter.

The latency factor parameter depends on the retrieval threshold, T, which varies substantially between ACT-R models. However, the following general relationship has been discovered: $F = 0.35e^T$ which means that the retrieval latency at threshold (when $A_i = T$) is approximately 0.35 seconds (Anderson et al., 2004). The full equation used by ACT-R to determine a chunk's activation takes into account several aspects, but its ba-

sis is the chunk's base-level activation. The base level activation B_i reflects the recency and frequency of use of the chunk, and is calculated by:

$$B_i = ln(\sum_{j=1}^{n} t_j^{-d}) + \beta_i$$

n: The number of presentations for chunk i.

t_j: The time since the j_{th} presentation.

d: The decay parameter. Standard this one is set at 0.5 (Anderson et al., 2004).

β_i: A constant offset.

When we assume that people are able to unconsciously employ a kind of utility analysis (which includes having implicit knowledge about what they can remember, see (Gray et al., 2006)) and adopt these interactive routines that minimize performance costs, we expect to find differences in behavior between the two cost conditions introduced.

5.3.3 Experiment

Sixteen first year AI students, aged between 17 and 24 years, participated in the experiment. The experiment's duration was approximate 30 minutes and participants received theoretically from 1 to 10 euro for participation, depending on their performance. In the experiment a 2-factor, between subjects design was used, with *costs* varied between participants. In condition 1, the costs of pressing a button were relatively high compared to the costs of an error, while in condition 2 the opposite was the case. For an overview of exact costs, see Table 5.5.

Participants started by reading a written instruction on how to perform the experiment and the costs of errors, time and sensing. Next, a practice task was given to familiarize them with the task and the costs. This task was similar to the main task, but in order to keep a low interference, color and shapes of objects were altered. Furthermore, the bin in which often or rarely encountered objects belonged and the order in which the objects were presented was altered.

Data Analysis

For data analysis we first calculated for each bin and at each trial the expected activation value of the participant's knowledge concerning the color, shape and the whole object (all) that would fit in the bin. For this we used the ACT-R formula with a standard decay of 0.5 and an offset of 0. As 'presentations' we counted the display of bin information due to button use, and the display of the correct bin at the end of each trial. Next, these activation values were used for regression analysis across participants for each

trial. Trials where the activation was 0 (e.g., the object had not been presented before) were excluded from analysis.

Univariate variance analysis was used to check for differences between the two conditions. For the difference between color and shape, a repeated measure ANOVA was conducted, using the Huyn-Feldt correction. For all analysis, trials with a RT exceeding 8000ms were excluded.

Experimental Results

Over all the participants, the percentage correct ranged from 30 to 97 percent; the average percentage correct was 61 ($SD = 21$). The number of times a participant chose a bin immediately ranged from 5 to 34; the average was 24.44 ($SD = 7.87$). So overall, there was a wide variety in the participant's behavior. As an illustration, Table 5.7 shows the data of one participant (participant 9). For each trial the participant's reaction time (**RT**, the time it takes to choose a bin or button), action (**Sense**, what feature was sensed) and performance (**Correct**, which feature was correct) are shown.

Table 5.7: Experiment data of participant number 9.

Trial	Object	RT	Sense	Correct
1	BT	1370	Shape	Shape
2	RC	960	Color+Shape	Color+Shape
3	RS	1106	Color+Shape	Color+Shape
4	BC	1564	None	Color
5	RC	1791	None	None
6	RT	923	None	Shape
7	YS	1222	Color+Shape	Color+Shape
8	RC	1399	None	Color+Shape
9	BS	2212	None	None
10	RC	800	None	Color+Shape
11	RS	1766	None	None
12	YC	2048	None	None
13	BC	2783	None	Shape
14	RS	1251	None	None
15	RC	804	None	Color+Shape
16	RC	1962	None	Color+Shape
17	YT	564	Color+Shape	Color+Shape
18	RS	930	None	Color+Shape
19	BC	5168	None	Shape
20	BS	1158	None	None
21	YC	2315	None	Color
22	RT	1390	Color+Shape	Color+Shape

Table 5.7 – Continued

23	BC	2044	None	Color
24	RC	672	None	Color
25	RS	1338	None	Color+Shape
26	BT	1479	None	Color
27	BC	2479	None	None
28	YS	2415	None	None
29	RC	3315	None	Color+Shape
30	RC	1154	None	Color+Shape
31	BS	2023	None	Shape
32	RT	1250	Color+Shape	Color+Shape
33	BC	3060	None	None
34	RS	1999	None	Color+Shape
35	YC	974	Color+Shape	Color+Shape
36	BS	5372	None	Shape

	Independent Variables								
Dependent	**First_Choice**			**RT_First**			**RT_Bin**		
Variables	p	R^2	r	p	R^2	r	p	R^2	r
Act-Color	0.002	0.27	0.52	0.000	0.48	-0.69	0.000	0.46	-0.68
Act-Shape	0.000	0.38	0.62	0.000	0.36	-0.60	0.000	0.35	-0.59
Act-All	0.001	0.35	0.59	0.000	0.53	-0.73	0.000	0.59	-0.73
Dependent	**Sense_Feature**			**Correct_Bin**					
Variables	p	R^2	r	p	R^2	r			
Act-Color	0.002	0.28	-0.53	0.004	0.24	0.49			
Act-Shape	0.000	0.40	-0.63	0.006	0.22	0.46			
Act-All	0.000	0.43	-0.65	0.002	0.32	0.57			

Table 5.8: Results of Regression Analysis.

The results of the linear regression analysis are shown in Table 5.8. The R^2 (explained variance), r(correlation) and p-values are given for each analysis. The results show that the activation value of color, shape and the whole object was successful in predicting a number of variables, confirming that the ACT-R theory correctly captures how human memory operates. For example, the Blue Circle in trial 13 was an object that was only shown 1 time before. Therefore, the activation value of this object was low (-1.77 on average), which coincided with the low mean percentage correct when participants immediately chose a bin (0.18).

First_Choice (the number of participants who chose a bin immediately) is positively dependent on activation value: as activation increased, First_Choice increased. Furthermore **RT** (reaction time) was examined: RT when the object is shown for the first time (‘**RT_First**’) and the time from the presentation of the object to the moment the bin was chosen (‘**RT_Bin**’). Both RT’s are dependent on the activations: RT decreased when activation value increased.

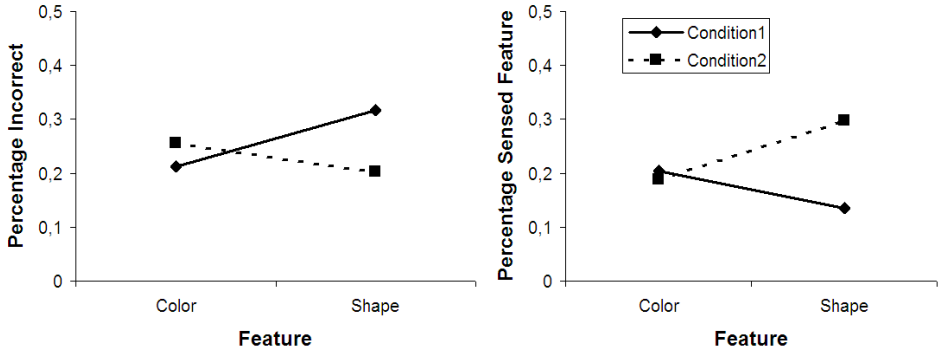


Figure 5.8: Interaction between feature and condition on percentage incorrect and percentage sensed feature.

In addition, the percentage of correct classifications concerning color, shape and all was found to be positively dependent on the activation of color, shape and all, see Table 5.8. When the activation increased, the percentage correct increased as well. The number of times a specific feature was sensed ('**Sense_Feature**') for color, shape or all decreased as the activation value of that feature increased.

Figure 5.8 shows the results of the ANOVA on the interaction between condition and feature. A trend is revealed when looking at the percentage incorrect. In condition 1 participants' percentage incorrect of shape ($M = 0.32$, $SD = 0.15$) was higher than that of color ($M = 0.21$, $SD = 0.15$; $F(1,7) = 6.81$, $p < 0.04$). For participants in condition 2 no such difference was found. An interaction is found between condition and feature on the number of times participants sensed a feature. For participants in condition 2 a trend was revealed, which showed that the percentage of sensed shape ($M = 0.30$, $SD = 0.27$) was higher than the percentage of sensed color ($M = 0.19$, $SD = 0.28$; $F(1,7) = 4.37$, $p < 0.1$). For participants in condition 1 no significant difference was found between the percentage of sensed color and the percentage of sensed shape.

Other than these interactions, no differences were found between the two conditions. This indicates that participants did not always make a rational decision, otherwise we would have expected to find more variety, e.g., in the total number of times features were sensed. Support for the thesis that humans instead rely on a prefixed strategy is found in the data, e.g., although participant 2 and 12 had the same condition, participant 2 always chose to acquire unknown information from the world (by pressing the *all* button 'l'), whereas participant 12 always attempted to retrieve it from memory (never pressed any button). Other support for relying on prefixed strategies can be found in the description of their approach by the subjects themselves. Table 5.9 gives an overview. As can be

seen, the responses are very diverse, and do not always seem to reflect rational decision-making.

Table 5.9: Strategies as described by the participants themselves.

Response	**Strategy** (literally translated descriptions)
1	After I knew the red circle, and a red shape was asked I choose color. Therefore I had 50% chance on a correct guess. After I stored some in my memory I was able to make a right choice for color or shape more often, thus leaving only one option because I knew that the others were different.
2	I first looked for the shapes and then I guessed, until you knew the colors many red at the left side, so use that to primary
3	Here I more often pressed 'everything'
4	Guessing and memorizing, two colors (red and blue, then you also know where the yellow ones are)
5	Only after some time I got clear where the shapes were, in the beginning I had to guess and the more I saw it, the better I could guess right
6	Choosing for shape and color ('l') if unsure or unknown, else answer
7	The first trials requesting both shape and color, afterwards only shape or color in order to have a chance of a half on a right guess (assuming that I still knew the requested objects). Initially this went quite well, but my memory is a strainer, so eventually remembering shapes and colors didn't went very well :-)
8	My strategy was to first look at the color and then choosing between the options until I had all colors and then came the shapes by experience, again 5,45 earned.
9	My strategy was to take another bin that had the same color. The right answer became visible and could be recorded.
10	My strategy was to show everyting in the beginning. In this way I was able to see the requested bin. In this way I was able to learn quickly where what was. This worked, but had as consequence that in case of a mistake, this often was 'both wrong', because I remembered the location per combination.
11	First using the L-key (show both), until you knew more or less where is what, then gradually less the L-key and guessing the location.
12	Guessing in the beginning and remembering.
13	A mix of only color and only shape. First guessing and later you can use logical reasoning to see where is which shape. (Thus, first color, then you know e.g. that blue is in 9, then shape in the second trial and then you see that square is in 9). If you do this reasonably well you can for surely earn 7 euros.

Discussion of Experimental Results

Overall, the results show that people's decision to acquire information from the world or from memory correlates with the activation of that information in memory following ACT-R's base-level activation formula, and is thus dependent on the frequency and recency of using that information.

A difference is found between color and shape, in that shape appears more difficult to retrieve from memory than color. This is shown by the fact that when people retrieve information from memory, the chance of making a mistake concerning shape is higher than the chance of making a mistake concerning color, see Figure 1. When the costs of acquiring information from the world are relatively low, this difference disappears as in such a situation people request shape (button 'k') more than color (button 'j').

No other differences are found between condition 1 and 2 when looking at the participant's reaction times or actions (sense or choose bin). This indicates that the decision to rely on information in-the-world versus information in-the-head is not influenced by the specific costs of acquiring that information. Rather it seems that people make a decision based on their own (pre-)specified strategy.

This finding does not necessary conflict the hypothesis that humans optimize their interactive routines to minimize performance costs. Gray and Fu (2004) and Gray et al. (2006) only consider performance costs measured in time, and argue that humans are evolved to conserve the resource of time. For the task presented in this paper performance costs are a combination of time and money costs, and it is conceivable that humans are not good in taking into account the money costs of actions. Since the time costs of actions do not alter between the two conditions, this might explain that no more differences can be found between them. On the other hand, people definitely attempt to optimize their performance based on time and money costs. When this would not be the case and they would only optimize the time costs, they would never press a button.

5.3.4 Task Model

As mentioned in the introduction, our research goal is the development of methods and techniques that will enable intelligent agents to display human-like behavior which might be rational, but often is not. For this goal we previously developed a memory model enabling rational as well as biased reasoning (Heuvelink et al., 2008a). This model was implemented in SWI-Prolog (Wielemaker, 2003), and incorporates ACT-R's base-level activation formula for declarative knowledge in memory. In this paper we take that model as basis for the development of a task specific model capable of executing the task previously introduced: http://human-ambience.few.vu.nl/docs/CogSci-IIAModel.pl.

Execution Loop

A run of the model starts by requesting the start of the task for a specific condition and individual. This sets the current time and trial at 0 and starts the model's execution cycle by calling the `scenario_loop` clause:

```
scenario_loop :-
    current_trial(Cond, Ind, T1),
    retract(current_trial(T1)),
    T2 is T1 + 1,
    assert(current_trial(Cond, Ind, T2)),
    sense_and_form_goal,
    determine_strategy_for_goal,
    sense_and_store_result,
    scenario_end.
```

The last predicate of the scenario loop, `scenario_end`, ensures that as long as the `current_trial` is not equal to 36, `scenario_loop` keeps being called.

In `sense_and_form_goal`, the model observes the presented object, which takes time `T` as specified by `time_required_to_ observe_goal_object(Cond, Ind, T)` and stores the observed object as `goal_at_trial(classify_object(C, S), T)`.

In `determine_strategy_for_goal`, the model executes a specific strategy on which we elaborate in the next section.

In `sense_and_store_result`, the model observes the correct result and stores this in its memory as `belief(color_shape, B, [C, S], T, passive_sense_result, 1.0)`, which denotes the belief that in bin `B` (1-9) color `C` and shape `S` fit. The `T` denotes the time at which this belief held, `passive_sense_result` the source of the belief and `1.0` its certainty. The fact that each belief receives a time, source, and certainty label is adapted from the memory model. In addition, the belief receives a so-called `impression_value`, which forms the constant offset of its activation level. The level of this impression value depends on whether the model chose the correct or a wrong bin. In case it was correct, the impression value `V` is equal to the `impression_value_correct_result(Cond, Ind, V)`, otherwise to the `impression_value_false_result(Cond, Ind, V)`.

After any belief is stored a process of the memory model becomes active called `deduce_abstract_belief_from_belief(B)`. This process deduces specific abstractions from stored beliefs with as main feature the deduction of semantic knowledge out of the episodic knowledge formed by beliefs with the introduced time, source and certainty labels. For the current task model, abstract beliefs are formed that abstract away

from the T , S and C of the specific beliefs. It are exactly those abstractions that over time have multiple 'presentations' and therefore receive a high activation value.

In addition, from beliefs about the color and shape that fit in bins, beliefs are abstracted that only capture knowledge about the color, or about the shape that fits in a specific bin. This separate storage of that information is inspired by the literature that claims that features are stored independently in memory, although they are bounded by their spatial location, in our case the bins (Johnson et al., 2002).

Heuristic Strategies

Gray and Fu (2004) state that the cost-benefit considerations for interactive routines only provide a soft constraint on their selection as they may be overridden by deliberately adopted top-down strategies. We have two indications that this might have happened with participants in our task: 1) the statistical analysis did not indicate significant differences between behavior on the two task conditions which would be expected when costs-benefits of actions would have been considered; 2) participants explicitly answered the open question 'What strategy did you follow' with answers like: 'When I did not know the correct answer I would pick a bin of which I knew it had the correct color.' (see response 9 in Table 5.9).

Based on logical reasoning and inspired by the participants' answers, we came up with 37 possible strategies participants could follow. The strategies mainly differ in the number of retrieval actions humans are willing to execute, and the order in which they do so. The strategies can be classified as embedding 1 to 3 retrieval steps. There is also the possibility of an extra security check, to see whether the bin selected to be chosen is not in conflict with the given object (e.g., when checked, it turns out that the shape of the selected bin can be retrieved and conflicts that of the object). Possible actions that can be taken after one of the retrieval steps are:

- choose a random bin (a)
- choose a random bin with security check (b)
- press show color/shape button, then choose random one of the three presented bins with security check. (c/d)
- press show all button, then choose that bin. (e)

Figure 5.9 summarizes all strategies. In the **first retrieval step** it is tried to retrieve the bin that matches the whole object which is presented. When retrieval is unsuccessful, any one of the actions a, b, c, d and e can be taken, which results respectively in strategies 1, 2, 3, 4, 5.

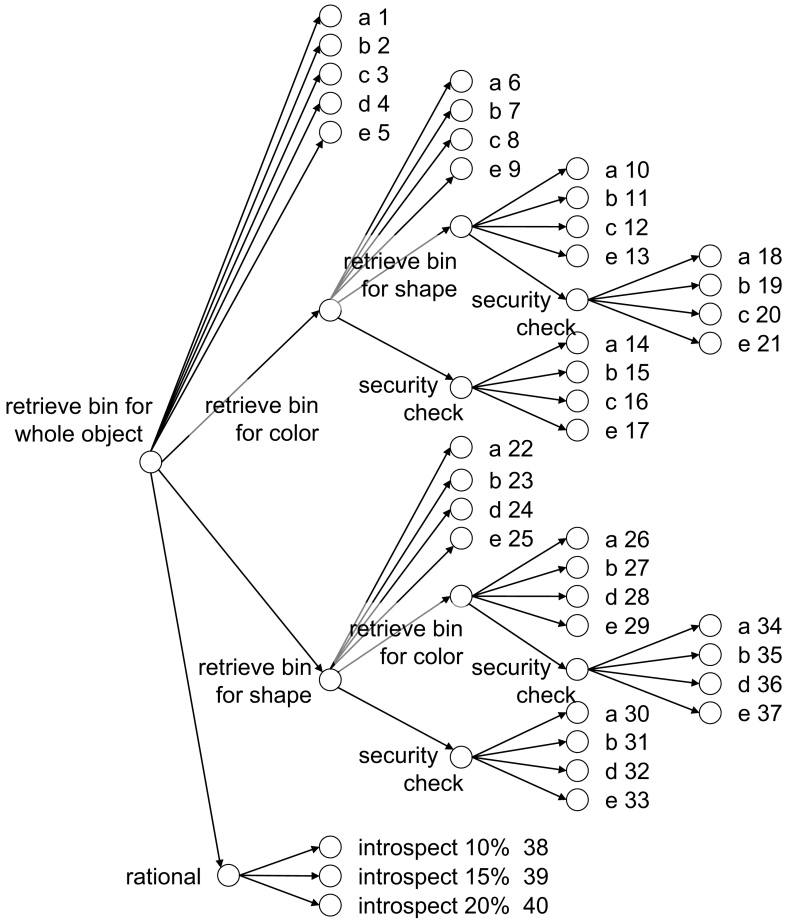


Figure 5.9: Schematic overview of all strategies.

Instead of directly choosing an action after unsuccessful retrieval of an object, a participant can make a **second retrieval step** to retrieve a bin of which either the color or the shape fits that of the object. If it is possible to retrieve the specific feature, that bin will be chosen. If it is not possible to retrieve the feature, again a specific action will be taken. For strategy 6 to 9 and 14 to 17, action a, b, c and e will be taken directly after an unsuccessful attempt to retrieve color. The difference between strategies 6 to 9 and 14 to 17 is that the latter, in case color can be retrieved, perform a security check. Strategy 22 to 25 and 30 to 33 are the same, but attempt to retrieve shape instead of color, and actions a, b, d and e are taken.

There is also the possibility of a **third retrieval step** after retrieving color or shape.

That is, if color can not be retrieved, in such strategies people will first try to retrieve shape before taking an action. Strategy 10 to 13 first try to retrieve color, then try to retrieve shape. Strategies 18 to 21 do the same, but with an extra security check. Actions a, b, c and e are taken when retrieving is unsuccessful. Strategy 26 to 29 first try to retrieve shape, then try to retrieve color (strategy 34 to 37 with an extra security check). Actions a, b, d and e are taken with unsuccessful retrieval.

In addition to the 37 strategies just introduced, we also implemented the rational strategy and included it as strategy 38-40. These strategies were equal in their determination of the expected costs of each action, but varied in the time it took them to introspect the activation values of the beliefs. This took them respectively 10, 15 and 20% of the time that it would take to actually retrieve the belief inspected.

In case a strategy would lead to action a: choice of a random bin, any of the nine bins could be chosen. Although the model would select one of these options, all of them were denoted as possible chosen bins. Similarly for action b: all the random bins of which no conflicting information could be retrieved were denoted as possible chosen bins.

When action c, d, or e was selected, the bins that fitted the requested information were revealed, and this knowledge was stored. The impression value of the stored information dependent on the feature sensed, as denoted by `impression_value_sense_color/shape/all_bin` respectively. Also from these specific beliefs about the color, shape or color-shape of bins, abstract beliefs were deduced. Next, (one of) the revealed, non conflicting bin(s) was chosen, and the possible bins that could have been chosen denoted.

5.3.5 Parameter Fitting

The model as described above contains a large number of parameters. Each specific parameter setting will result in different behavior of the model. To answer the question to what extent the model can correctly describe human behavior, we performed a technical experiment with which we tried to find parameter settings for which the model displays behavior close to that of a participant.

Unfortunately, due to the large number of parameters, we were unable to fit them all. For the current research we focused on fitting the various strategies as well as the specific parameters that influence the storage and retrieval of beliefs. This means that the parameters that influence the time to sense information and to execute actions were fixed. In specific, we fixed the following parameters to the given values, based on inspection of the empirical human data:

```
time_required_to_observe_goal_object(_,_, 0.3)
```

```
time_required_to_press_button(_,_, 0.4)
time_required_to_press_bin(_,_, 0.7)
```

The technical experiment has been performed as follows. First, we analyzed the empirical human data further to find realistic ranges for the parameters in the model. This resulted in the following parameters that were run:

```
impression_value_sense_color_bin:  0.0, 0.1, 0.2
impression_value_sense_shape_bin:  0.0, 0.1, 0.2
impression_value_sense_all_bin:
    impression_value_sense_shape_bin + 0.0, 0.2
impression_value_correct_result:  0.0, 0.1, 0.2
impression_value_false_result:
    impression_value_correct_result + 0.0, 0.2
retrieval_threshold:  -1.0, -0.7, -0.4, -0.1, 0.2, 0.5
strategy:  1, 2, , 39, 40
```

As can be seen, we decided to separately denote the `impression_value` that a belief about the color, shape, or color and shape of a bin would receive after sensing color, shape or both, respectively. This way it is possible for a possibly existing difference in how well color and shape are remembered to show up. Due to the large number of parameters already present we decided not to parameterize the impression value given to the abstract beliefs about the color or shape of a bin that were deduced from beliefs about its color and shape. Therefore, differences between the storage of color and shape could only show up in case the model selects the sense-color and sense-shape button.

In addition, we decided to make the `impression_value` of `sense_all_bin` dependent on (equal or larger to) the `sense_shape_bin`, and the `impression_value` of a `false_result` dependent on (equal or larger to) that of a `correct_result` . The reason we did this is that the impression value denotes the amount of attention paid to the information to be remembered. We gathered it illogical that one out of three features would receive more attention than one out of two, or that a correct, probably expected, result would receive more attention than a false, important to remember, result.

Next, we used the model to run simulations for all the possible combination of the introduced parameter settings. This meant that we ran the model 27,864 times (twice for strategies 38-40 due to the influence of the task condition), each time giving the model the same 36 objects to classify as were given to the participants. For all parameter settings and at each trial the following information was logged: the action executed by the model (sense-color, sense-shape, sense-all or none), its reaction time (RT, the time until the button, or in case of 'none' the time until a bin number was pressed), and the possible bins the model could have chosen.

Subsequently, we compared each participant with the 27,864 simulation results. To do this in a structured way, we developed a distance measure that calculates for each trial a distance between the model data and the data of the participant. For this, we first calculate a distance value for each aspect, in case of reaction time RT by the following formula:

$$distance\_RT = |human\_RT - model\_RT| / (2 * SD)$$

SD: the standard deviation of the human reaction times.

For the $chosen\_bin$, the distance was 0 when the human had chosen a bin which was one of possible the bins the model could have chosen, and 1 otherwise. For $action$, the distance was calculated according to Table 5.10.

Action	Color	Shape	Color+Shape	None
Color	0	1	0.5	0.5
Shape	1	0	0.5	0.5
Color+Shape	0.5	0.5	0	1
None	0.5	0.5	1	0

Table 5.10: The distance measure for actions.

For the overall distance measure we decided to let the similarity between the human action and model action have the strongest influence, followed by the similarity of the bin in which the object is classified, while the reaction time only has a slight influence:

$$distance = (6 * distance\_action + 2 * distance\_chosen\_bin + distance\_RT)/9$$

The reason for this measure was that the use of different strategies, which is the focus of this paper, mainly shows up in the action choices. In addition, we did not expect to find very good fittings for the reaction times due to the fixing of the `time_required_to` `parameters` that largely determine the model's reaction times.

Results Parameter Fitting

The results of the parameter fitting for four different participants will now be discussed. Although this is not yet a thorough validation of the model, it provides evidence for the feasibility of the model. The subjects, two for each condition, were selected based on typical behavior patterns: participant 2 (condition 2) almost always requested information, participant 7 (condition 1) almost never did. Participant 9 (condition 1) and 10 (condition 2) were chosen because they seemed to perform rational behavior (more sensing in the beginning, less sensing at the end). Table 5.11 shows the actions of the four participants.

Table 5.11: Sense actions of the 4 participants.

Trial	Object	PP2	PP7	PP9	PP10
1	BT	Color+Shape	Color	Shape	Color
2	RC	Color+Shape	Color	Color+Shape	Shape
3	RS	Color+Shape	None	Color+Shape	Shape
4	BC	Color+Shape	None	None	Shape
5	RC	Color+Shape	None	None	None
6	RT	Color+Shape	None	None	Color
7	YS	Color+Shape	None	Color+Shape	Shape
8	RC	Color+Shape	None	None	None
9	BS	None	None	None	None
10	RC	None	None	None	None
11	RS	Color+Shape	None	None	None
12	YC	Color+Shape	None	None	Shape
13	BC	Color+Shape	None	None	None
14	RS	Color+Shape	None	None	None
15	RC	Color+Shape	None	None	None
16	RC	None	None	None	None
17	YT	Color+Shape	None	Color+Shape	Shape
18	RS	Color+Shape	None	None	None
19	BC	Color+Shape	None	None	Shape
20	BS	Color+Shape	None	None	None
21	YC	Color+Shape	None	None	None
22	RT	Color+Shape	None	Color+Shape	Shape
23	BC	Color+Shape	None	None	Color
24	RC	Color+Shape	None	None	None
25	RS	Color+Shape	None	None	None
26	BT	Color+Shape	None	None	Color
27	BC	Color+Shape	None	None	None
28	YS	Color+Shape	None	None	Shape
29	RC	Color+Shape	None	None	None
30	RC	None	None	None	None
31	BS	Color+Shape	None	None	None
32	RT	Color+Shape	None	Color+Shape	Color
33	BC	Color+Shape	None	None	Shape
34	RS	Color+Shape	None	None	None
35	YC	Color+Shape	None	Color+Shape	Shape
36	BS	None	None	None	None

The lowest distance values of subjects 2, 7, 9 and 10 are 5.242, 2.105, 5.555 and 6.340 respectively. For all participants the settings with distance values that lie within 1% of this lowest distance value were analyzed. This resulted in only 1 setting for participant 10, but 7, 18, and 16 different settings for subjects 2, 7, and 9 respectively. We found

that the parameters for `strategy` and `retrieval_threshold` were equal across the settings per participant, but that the `impression_values` strongly fluctuated per setting. This, however, is not surprising as differences stemming from the setting for, e.g., `impression_value_sense_color_bin`, only show up in case this sense action is actually selected.

The strategy parameter that fits participant 2 is strategy 5, with a retrieval threshold of 0.5. This strategy entails that when an object can not be retrieved from memory, its position will be requested. Because the model's retrieval threshold is very high (0.5) the objects' activation values frequently lie below the retrieval threshold. Therefore, the model is often unable to retrieve the presented object, and thus often (30x) requests information. This represents the choices of participant 2, who 31 times pressed button 'l': sense-all. Looking at Table 5.11, it appears that the participant could only remember the frequently presented red circle, and the blue square. Analysis of the best matching setting pointed out that action of subject 2 indeed correlates with model action ($r = 0.47$, $p < 0.01$). Reaction time of subject 2 does not correlate with model reaction time.

Strategy 30 and a retrieval threshold of 0.5 fit best with participant 7. This strategy often results in directly choosing a bin as when shape can not be retrieved, a random bin is chosen. This is apparent in participant 7, who only pressed a button at the first two trials. The relatively low distance (2.103) follows from the fact that when the model chooses a random bin, the bin chosen by the participant always matches the possible chosen bins of the model. No significant correlations were found between model RT and human RT and between model action and human action. This is partly due to the fact that the values of model RT and model action varied little and not at all, respectively.

Participant 9 fits best with strategy 39 and a retrieval threshold of -0.1. Strategy 39 is a rational strategy taking the costs of acquiring information from the world and from memory into account. Since this participant had been assigned the condition in which the button costs are high and penalties low, such a strategy would result in a pattern that the only time information will be acquired from the world is when the chance or error is really large, e.g., for an object rarely encountered. This behavior is indeed shown in participant 9, see Table 5.7. For example, on trial 17 a Yellow Triangle was presented, an object which was never presented before, and that was one of the few (7) trials the participant decided to press the sense-all button. Further analysis revealed a significant correlation between human action and model action ($r = 0.68$, $p < 0.01$), but also between human RT and model RT ($r = 0.40$, $p < 0.02$).

Strategy 36 and a retrieval threshold of 0.2 fit best with participant 10. Strategy 36 is, contrary to our expectations, not a rational strategy. The strategy either results in choosing a bin (when either shape or color is known), or in sensing the shape (when

shape and color are both unknown or one of them conflicts). The choices of participant 10 reveal such a pattern as the participant's actions are mainly to directly choose a bin or to sense shape. This is confirmed by the significant correlation between human action and model action ($r = 0.61$, $p < 0.01$). In addition, a trend in correlation was found between human RT and model RT ($r = 0.31$, $p < 0.1$).

5.3.6 Discussion & Conclusion

The results show that it was possible to find parameter settings that matched reasonably well with the four investigated participants, especially on the executed actions. Reaction time proved to be a less optimal measurement for parameter fitting. This could be due to the fact that we set a fixed time to observe information, and to press a bin or a button for all participants. As reaction time is personal, such parameters need to be fitted as well.

We can also conclude that people adopt different strategies to decide whether to acquire information in-the-world versus information in-the-head. At this moment we think that many of our participants already had decided on how to act, instead of deciding this on-line. The descriptions of the strategies as listed in Table 5 support this hypothesis.

With hindsight knowledge, we can make a few critical remarks about our experimental setup and our model. First, the task that was given to the subjects was too complex with respect to the cost parameters. There were too many cost parameters, in addition to the fact that we used two types of costs (time and money). This made it difficult for the participants to do an accurate cost-benefit analysis, shown by the fact that we were not able to clearly distinguish an effect of the different cost conditions. It is interesting to find out whether this would be different for tasks that are less complex and only involve one type of costs. In such a situation the effect of costs of information acquisition actions and costs of errors can be studied more closely.

Second, it became clear that the setup of the task made it possible to choose a strategy that optimizes the utility over different trials. Some participants preferred to sense 'color' or 'shape' over 'color+shape' because the first two options revealed information about objects in three bins instead of information about an object in one bin (e.g., see response 13 in Table 5). As the rational strategies in our model do not take this into account, such strategies won't fit to a rational strategy in the model, although they actually are rational. This could also explain why the behavior of participant 10, which appeared rational, did not fit best with a rational strategy (see the previous section).

Third, we can conclude that we made a suboptimal choice in selecting the parameters to be fitted. Major parameter settings were fixed (time to observe information and time to execute actions) while it was attempted to fit others (impression-values of sensed information) that were of much less importance to task execution.

Fourth, it is a question whether our 'meta-model' for deriving the 37 strategies is correct, i.e., the idea that the heuristic strategies vary in the number (and order) of retrieval actions humans are willing to take to come to a decision. The rational strategy decides by (unconsciously) considering all the possible retrieval and sense actions and their effects at the same time. The heuristic strategies execute a more serial process; they execute a retrieval action, and then decide on what to do next, which could be further deliberation.

The modeling of these different approaches to decide on what to do resembles the work of Dickison and Taatgen (2007), who state that for complex tasks it may become impossible to model individual differences by parameter tuning. Instead, they propose that people differ in the control strategies they employ, and that these manifest themselves as different problem-solving strategies. These control strategies supposedly differ in the amount of top-down control exerted on behavior, opposed to this behavior being driven by bottom-up processes.

It could well be that people differ in the type of control they exert (with top-down control leading to more rational behavior) based on other individual differences, e.g. the capacity of their working memory (WM). Differences in WM capacity have been used to explain the differences between the task strategies selected by different humans under the same task circumstances, as by the same human under different circumstances (Beilock and Decaro, 2007). Given these findings, we think that our approach to capture varied human decision-making by modeling (heuristic) strategies that vary in the number of retrieval actions humans are willing to make, is a feasible one.

In future work, we would like to redo the experiments using the insights that are described above, i.e., using a simpler task with fewer cost parameters. In addition, we want to vary the various time-to-do-x parameters and to fit the model on these parameters as well. Moreover, we would like to extend the model so it does not execute a predetermined strategy, but on-line selects one, e.g., based on the available WM capacity. Furthermore, it might be useful to do a separate experiment with a simple task to further investigate the difference we found between the retrieval of color and shape.

Chapter 6

Cognitive Agent Capabilities

6.1 Introduction

In the previous chapters we modeled several cognitive capabilities in the form of components of cognitive agent models. We decided to follow a component-based approach to the modeling of cognitive agents to foster a structured, cost-effective agent-development method (Section 1.1.5). An important factor for making a component-based agent-development method cost-effective is the reuse of previously developed components. To support this, these components should be available in some repository that can be queried for relevant agent components. In addition, these existing components should be tagged with a proper description, so that they can be discovered for reuse.

In this chapter, we introduce a preliminary line of research that in the long run might support the tagging, thus querying, and thus reuse of cognitive agent components. The basic idea underlying the research is that cognitive agent components usually embed specific cognitive capabilities. From this it follows that it is useful to tag these components with a description of the cognitive agent capability they embed; advisable with a specification of the properties of that capability implementation. To support this we started the research presented here: the development of a Capability Description Framework (CaDeF) that can be used to analyze and describe the capabilities and accompanying properties of cognitive agent models.

Chapter overview

This chapter embeds a single paper (Section 6.2) in which we introduce our preliminary ideas for a Capability Description Framework (CaDeF). CaDeF has two functions, it 1)

defines a method for describing cognitive agent capabilities, and 2) provides definitions of generic, basic capabilities. The method proposes that capabilities are described by defining the properties of the entities that make up the capability, and that there exist three types of such entities: *means*, *processes on means*, and *control of processes on means*. For each of these entities *functional*, *system*, and *dynamic* properties can be defined. We demonstrate the use of this method by defining two well-known, generic cognitive agent capabilities. In addition, we demonstrate its use by describing specific implementations of these two capabilities in the existing cognitive agent BOA (Section 3.3).

Additional Remark

In the presented paper we use a semi-formal notation that combines aspects of set theory and predicate logic to clarify the ideas underlying CaDeF. This notation is only temporary; the ideas could have been formalized in another way. In order to develop a full formal framework additional research is required, at which time it also has to be decided whether an algebraic or logical approach is more suited.

Research Paper

6.2 CaDeF: Towards a Method for Describing Cognitive Agent Capabilities

Abstract

Reusing existing (parts of) cognitive agent models requires that it is specified which cognitive capabilities and properties each embeds. This is especially the case because capabilities can be realized in multiple ways, giving rise to capability variants. In this paper we introduce CaDeF, a framework to describe cognitive agent capabilities and properties. A capability is defined by the properties of its means, processes, and control of processes; a capability variant by specifying additional properties. We demonstrate the Capability Description Framework by defining two well-known, generic cognitive agent capabilities. Its applicability is shown by describing the implementation of (variants of) these capabilities in an existing agent.

This section is an unpublished paper:
Heuvelink, A., Mioch, T., and Doesburg, W. A. van. CaDeF: Towards a Method for Describing Cognitive Agent Capabilities.

6.2.1 Introduction

Our research concerns the development of cognitive agent models that, when implemented in software, can replace human role-players in training simulations and thus cut back the expenses of training. To ensure the cognitive agents do this, the costs of developing them should not be too high. This entails that it should not be so that for every new domain, task, or even scenario a new agent model has to be developed from scratch. Therefore, our research takes a component-based approach to the modeling of cognitive agents.

A prerequisite of a component-based design approach for cognitive agents is that one must be able to 1) design an agent model by specifying its capabilities and for each capability the specific properties for the task at hand, and 2) find existing components that support this specification. Unfortunately, there does not exist consensus on an approach to catalog cognitive agent capabilities and their properties, nor on the level at which these should be described. The focus of the research described in this paper is the development of a framework that can be used to analyze and describe the capabilities and accompanying properties of cognitive agent models. With our work we aim to support future component-based, cognitive agent modeling approaches that require structured, meaningful descriptions of components.

We start this paper with a short discussion on decomposing cognition, which is followed by an introduction of the ideas underlying the proposed Capability Description Framework (CaDeF). First, we elaborate on CaDeF's capability definition. Then, we explain this definition further by means of two example capabilities. Next, we illustrate the applicability of CaDeF to analyze and describe capabilities of existing cognitive agents. We conclude this paper with a short discussion about the proposed framework and future research plans.

6.2.2 Related Work

Since the advent of (cognitive) agents it has been a recurring topic which capabilities a cognitive agent can, and should, possess (Franklin and Graesser, 1997). One reason for this discussion is that when it is understood which capabilities are critical for the modeling of behavior in agents, cognitive architectures can be built that support these capabilities.

A basic assumption underlying cognitive architectures as well as component-based cognitive agent designs is that human cognition is modular. Much research within psychology, artificial intelligence (AI), and neuroscience subscribes to this modularity of mind (Bryson, 2005). Fodor (1983) makes a distinction between *horizontal* vs. *verti-*

cal modules that both can be used to decompose cognition. *Horizontal* decompositions are those which identify processes which underlie all of cognition, such as memory, attention, perception, and judgment. Such a decomposition is embedded in cognitive architectures. *Vertical* decompositions identify particular skills, such as mathematics and language, that each have their own characteristic processes of memory, attention, etc. AI research that develops agents for specific goals, e.g., online auctions, is likely to subscribe to the vertical decomposition when developing agent components.

The literature shows that a wide variety of agent capability classifications are possible: some are inspired by cognitive theories and capture generic, 'horizontal' capabilities (Langley et al., 2006; Gluck and Pew, 2005), others stem from practical design choices and capture more specific, 'vertical' capabilities (Fineberg, 1995; Padgham and Lambrix, 2005). The fact that agent models and models of cognition are engineered makes that they can vary widely on their level of description and the roles their internal components play. However, it can be useful to reuse any of these model parts; it should therefore be possible to label each of them with a proper description of the capability variants they embed.

6.2.3 Approach

This paper proposes a Capability Description Framework for cognitive agent models. With CaDeF we aim to be able to describe the entire range of (parts of) models that can be considered to embed a capability: from abstract, generic models to task specific, instantiated ones. In order to bring structure to this wide amount of possible capability descriptions we adopt the following approach: 1) we describe the generic abstract cognitive capabilities that can be found in cognitive agents; 2) we develop a method to extend these generic descriptions to capture specific capability variants; 3) we investigate how (variants of) capabilities relate to each other.

At this moment we take as the basis for the generic cognitive capabilities to be described by CaDeF the work of Langley et al. (2006) that examines the capabilities that a cognitive architecture can support. They divide these capabilities into nine main areas, see Table 6.1.

Generic capability descriptions should be extendable to capture specific capability variants. We subscribe to the existence of two variant types: horizontal variants are formed by amending a generic description to capture new capability properties that hold for the entire agent; vertical variants are specifications of a generic capability whose added properties only hold for a task-specific module.

In order to model a new agent using CaDeF or to describe an existing agent, a functional analysis of the cognitive agent should be done to identify the capabilities of that

Table 6.1: Capabilities of Cognitive Agents, from Langley et al. (2006)

1.	Recognition and Categorization
2.	Decision-Making and Choice
3.	Perception and Situation Assessment
4.	Prediction and Monitoring
5.	Problem Solving and Planning
6.	Reasoning and Belief Maintenance
7.	Execution and Action
8.	Interaction and Communication
9.	Remembering, Reflection and Learning

agent. When a certain part of an agent plays an identifiable functional role (e.g. choosing between various goals) that part embeds a specific agent capability. In order to describe that capability not all the details of the part need to be recorded as properties of the capability, but only those aspects that are typical for its functioning. CaDeF functions as a means to record the outcomes of such a functional analysis.

Capability Definition

We consider a capability as a part of an agent that, by itself, can generate meaningful behavior. Because our final concern is the computational modeling of behavior, it is proposed that for defining a capability one needs to define resources (i.e., capability-means), processes acting upon these resources (i.e., capability-processes) and a logic that controls the behavior of that process (i.e., a capability-control), semi-formal denoted by:

`A capability is a collection of Means, Processes operating on these Means, and a Control entity:`

$$Capability = (Means, Processes, Control)$$

As mentioned in the previous section, capability entities are identified based on their functionality: the way in which specific entities of an agent are viewed depends on the role they fulfill within a capability. For example, in the next section we explain how some entities that have the role of means within one capability fulfill the role of processes in another. By describing capabilities in this functional way, the great variety found in level of detail of the entities of capabilities does not pose any problem for its description.

We propose to define the generic capabilities, whose descriptions are part of CaDeF, by denoting the properties that must hold for their Means, the ones that must hold for the Processes, and the ones that must hold for the Control.

For the means M of generic capability Cap, properties Prop_{M} must hold:

$$Means\_of\_Capability\_has\_Properties(M, Cap, Prop_M)$$

For the process P of generic capability Cap, properties Prop_{P} must hold:

$$Process\_of\_Capability\_has\_Properties(P, Cap, Prop_P)$$

For the control C of generic capability Cap, properties Prop_{C} must hold:

$$Control\_of\_Capability\_has\_Properties(C, Cap, Prop_C)$$

Additionally, we propose three types of properties: functional properties, system properties, and dynamic properties.

Properties Prop_{X} is a collection of functional properties FP_{X}, system properties SP_{X}, and dynamic properties DP_{X}:

$$Prop_X = (FP_X, SP_X, DP_X)$$

Functional Properties describe properties the capability entity has that are relevant to its role in the capability. In CaDeF we abstract away from what properties exactly mean: we assume that it is possible to define this in a separate ontology of cognitive agent terms. In CaDeF we denote properties simply as ordered attribute value pairs: $\langle A, V \rangle$.

System Properties describe inherent properties of capability entities, for example their representation. In generic capability descriptions the values of these properties are often not specified, because they are implementation specific. The reason to include them in the capability description is because these properties will be instantiated in specific variants, in which case they are relevant to know with respect to reuse.

Dynamic Properties also describe inherent properties of capability entities, but their specific value is not given by the developer, but by the current, dynamic, circumstances. As such, the value of dynamic properties always depend on the other capability entities.

A generic capability can be specialized. Specialization of a generic capability can be done by adding new properties that hold for any task the agent might perform (i.e. horizontal specialization). Additionally specialization can be done by adding properties that only hold for certain skills or tasks of the agent (i.e. vertical specialization)

Specific capability $\text{Cap}_{\text{specific}}$ is a specification of the Generic capability $\text{Cap}_{\text{generic}}$:

$$Specific\_Variant\_Of(Cap_{specific}, Cap_{generic})$$

6.2.4 Capability Cases

In this section we describe the generic capabilities *reasoning* and *decision-making* using CaDeF. This illustrates how CaDeF defines such generic capabilities by specifying the properties that must hold for each of the three entity types introduced. At the end of this section we discuss how certain levels of capabilities can relate to each other, causing the confusion between capabilities and properties of agents as often encountered, e.g., in Langley et al. (2006).

Reasoning

We consider reasoning "a central cognitive activity that lets an agent augment its knowledge state" (Langley et al., 2006).

In the following, we propose a definition of the generic capability *reasoning* using CaDeF by specifying the properties we believe each reasoning capability minimally needs to have. Capitals denote variables (for readability often informed variable names are chosen).

For the means Knowledge of generic capability reasoning, properties $\texttt{Prop}_{\texttt{Knowledge}}$ must hold:

$Means\_of\_Capability\_has\_Properties(Knowledge, reasoning, Prop_{Knowledge})$

$Prop_{Knowledge} = (FP_{Knowledge}, SP_{Knowledge}, DP_{Knowledge})$

Functional properties $\texttt{FP}_{\texttt{Knowledge}}$ are that Knowledge is declarative and accessible:

$FP_{Knowledge} = (\langle declarative, yes\rangle, \langle accessible, yes\rangle)$

Declarativeness means that an agent in principle can access the Knowledge's content, while accessible denotes that the agent can assess its content at the current moment.

System properties $\texttt{SP}_{\texttt{Knowledge}}$ are that Knowledge has representation R:

$SP_{Knowledge} = (\langle representation, R\rangle)$

Dynamic properties $\texttt{DP}_{\texttt{Knowledge}}$ are not defined:

$DP_{Knowledge} = \emptyset$

With the given definitions we denote the minimal properties the entities that function as means for reasoning have to have. They might have additional properties, e.g. certainty, representing an agent's confidence about the knowledge.

$\text{Processes}_{\text{Cap}}$ are entities that operate on the $\text{Means}_{\text{Cap}}$. We propose the following definition for the processes of the generic reasoning capability:

For the process Rule of generic capability reasoning, properties $\text{Prop}_{\text{Rule}}$ must hold:

Process_of_Capability_has_Properties(Rule, reasoning, $\text{Prop}_{\text{Rule}}$)

$Prop_{Rule} = (FP_{Rule}, SP_{Rule}, DP_{Rule})$

Functional properties FP_{Rule} are that Rule is task specific:

$FP_{Rule} = (\langle task\_specific, yes \rangle)$

System properties SP_{Rule} are that Rule has representation R, and determinism D:

$SP_{Rule} = (\langle representation, R \rangle, \langle deterministic, D \rangle)$

Dynamic properties DP_{Rule} are that Rule has executability E:

$DP_{Rule} = (\langle executability, E \rangle)$

The executability of a process is determined at runtime and depends on the current means.

It is possible to identify many additional process properties that can be used to define specific variants, e.g., the cognitive costs of processes, their utility, or their required input. These optional properties can play an important role for the control of reasoning, as is described at the end of this section.

$\text{Control}_{\text{Cap}}$ in an entity that determines which of the $\text{Processes}_{\text{Cap}}$ may become active. We propose the following definition for the generic control of reasoning:

For the control C of generic capability reasoning, properties Prop_{C} must hold:

Control_of_Capability_has_Properties(C, reasoning, Prop_C)

$Prop_C = (FP_C, SP_C, DP_C)$

Functional properties FP_{C} are that C is not task specific and that it always considers the processes' executability:

$FP_C = (\langle task\_specific, no \rangle, \langle considers\_executability, yes \rangle)$

That the control is task unspecific does not mean that for a specific task a specific type of control might not be more suited than another. But usually it is assumed that a single structure controls all reasoning, independent of the task.

System properties SP_{C} are that C has representation R, and loop type L:

$SP_C = (\langle representation, R \rangle, \langle type\_of\_loop, L \rangle)$

The type of loop can be either *single* or *multiple*, dependent on whether the control process can only execute once while executed the reasoning capability, or several times.

Dynamic properties DP_C are that C activates N active reasoning processes:

$DP_C = (\langle number\_of\_active\_reasoning\_processes, N \rangle)$

An additional property of control might be that it takes a certain constraint into account to determine which process to execute. For example, there can be a limited number of knowledge rules that may fire, or a maximal amount of processing costs these rules may have. To facilitate this the control may have as additional property that it take properties of processes into account besides their executability. At the end of this section we discuss such a specific control variant.

Decision-Making

We consider decision-making as "the ability to select among alternatives" (Langley et al., 2006). Here we describe our CaDeF definition of decision-making succinctly, in the next section we illustrate it further by means of an example.

For the means Alternative of generic capability decision-making, properties $Prop_{Alt}$ must hold:

$Means\_of\_Capability\_has\_Properties(Alternative, decision\text{-}making, (FP_{Alt}, SP_{Alt}, DP_{Alt}))$

Functional properties FP_{Alt} are that an Alternative is declarative and accessible, and has an additional Aspect A:

$FP_{Alt} = (\langle declarativeness, true \rangle, \langle accessibility, true \rangle, \langle Aspect, A \rangle)$

System properties SP_{Alt} are that an Alternative has representation R:

$SP_{Alt} = (\langle representation, R \rangle)$

Dynamic properties DP_{Alt} are that an Alternative has an evaluation score E:

$DP_{Alt} = (\langle evaluation\_score, E \rangle)$

Each alternative must have an additional Aspect A because based on this aspect the capability determines an evaluation score for it on which it basis its final decision.

Three different decision-making processes are identified; they specify three necessary steps within decision-making. The processes are labeled *Determination-of-Options (DO)*, *Evaluation-of-Options (EO)*, and *Selection-of-Options (SO)*. None of these processes has any required dynamic properties:

$DP_{DO} = DP_{FO} = DP_{SO} = \emptyset$

However, the processes differ in their functional and system properties. The first process *Determination-of-Options* is a structure that determines which of the Alternatives are currently actual options by taking a constraint C into account. For example, if a decision has to be made about which goal to attend to, this step might determine which goals are currently active and could therefore be actually attended to.

Functional properties FP_{DO} are that DO is task specific and considers a constraint:

$$FP_{DO} = (\langle task\_specific, yes\rangle, \langle considers\_constraint, yes\rangle)$$

System properties SP_{DO} are that DO has representation R and takes constraint C into account:

$$SP_{DO} = (\langle representation, R\rangle, \langle takes\_into\_account\_constraint, C\rangle)$$

The second process *Evaluation-of-Options* is a structure that determines evaluation scores E for the Alternatives based on one or more aspects A of them. This structure can embed an *injective* or a *non injective* function, which means that an option always receives a unique or possibly shared score respectively. For example, each active goal might receive a relevancy value based on their expected output.

Functional properties FP_{EO} are that EO is task specific and considers an aspect:

$$FP_{EO} = (\langle task\_specific, yes\rangle, \langle considers\_aspect, yes\rangle)$$

System properties SP_{EO} are that EO has representation R, outputs evaluation type T, and bases its evaluation on Alternatives' aspect A:

$$SP_{EO} = (\langle representation, R\rangle, \langle evaluation\_type, T\rangle, \langle takes\_into\_account\_aspect, Aspect\rangle)$$

The third process *Selection-of-Options* is a structure that determines which of the Alternatives are selected and basis this on their evaluation score E taking a constraint into account. This is based on the assumption that decision-making always involves some kind of decision: selecting randomly is not considered decision-making. For example, this process may select the goal which has the highest relevancy.

Functional properties FP_{SO} are that SO is task specific, and considers a constraint and the evaluation scores of the Alternatives:

$$FP_{SO} = (\langle task\_specific, yes\rangle, \langle considers\_constraint, yes\rangle, \langle considers\_evaluation\_score, yes\rangle)$$

System properties SP_{SO} are that SO has representation R and takes constraint C into account:

$SP_{SO} = (\langle representation, R\rangle, \langle takes\_into\_account\_constraint, C\rangle)$

Control C of the decision-making capability determines which of the three decision-making processes may execute.

Functional properties FP_C are that C is not task specific, and that it has a fixed process order:

$FP_C = (\langle task\_specific, no\rangle, \langle fixed\_process\_order, true\rangle)$

Independent of the task, the three decision-making capability processes are executed in the same order, namely first *DO*, then *EO*, and last *SO*.

System properties SP_C are that C has representation R and operates in a single loop:

$SP_C = (\langle representation, R\rangle, \langle type\_of\_loop, single\rangle)$

To make a decision, the control loop just has to execute once.

There are no dynamic properties DP_C: $DP_C = \emptyset$

Combining Capabilities

Previously we introduced CaDeF's definition of the generic reasoning capability. We mentioned that in addition to the specified required properties additional properties might hold for specific capability variants, e.g., the control might take constraints into account for deciding which of the executable reasoning processes may fire. When there exist such a constraint, e.g., on the number of Rules that may fire, a selection has to be made among the executable ones. Interestingly, a possible selection process is the decision-making capability.

When the control of reasoning is based on decision-making, *the processes of reasoning are interpreted as means for decision-making*. For this, the properties of the reasoning processes are expanded with the properties of the decision-making means. This entails that reasoning processes should be declarative and accessible. In addition, they should have an additional (functional or dynamic) property besides their dynamic property 'executability' that can function as aspect to base the decision on, and they receive as new additional dynamic property an evaluation score.

The processes and control of decision-making are comprised within the reasoning control. This entails, e.g., that the latter now has the additional functional property that it takes a constraint, and a processes' aspect into account.

With this section we want to stress that different capabilities can be embedded in each other and that the properties of the capability levels depend on each other. For example, when reasoning embeds decision-making, the reasoning processes should be specified declaratively and have an additional property that can be used to base a decision on.

6.2.5 Applying CaDeF to a Pre-Existing Agent

In this section, CaDeF is used to describe two capabilities of a previously developed cognitive agent named BOA (Both and Heuvelink, 2007), which is implemented in ACT-R (Anderson and Lebiere, 1998). BOA is developed to perform the so-called tactical picture compilation task, which consists of gathering and integrating information from a screen about radar contacts and classifying these contacts. In addition to the information from the screen, BOA can send a helicopter to gain visual information about contacts. In the following, we describe the horizontal variant of the reasoning capability embedded in BOA and a vertical decision-making variant.

Reasoning

The major part of the task execution by BOA consists of reasoning about its beliefs using its domain knowledge in order to deduce new beliefs. For BOA, the $\text{Means}_{\text{reasoning}}$ are the beliefs the agent holds. These beliefs are implemented, and thus represented, in the declarative memory of ACT-R as *chunks*, and are therefore declarative. BOA is based on the belief framework developed in Heuvelink (2007), which causes all beliefs to have as additional properties a certainty level, a source label, and a time stamp.

In ACT-R, each chunk automatically receives an activation level that determines how available it is. ACT-R limits the accessibility of the means for reasoning: only the chunk with the highest activation that matches a retrieval request can be retrieved. However, for BOA it is required that certain types of belief can be retrieved independent of their activation. Because this is impossible to do within ACT-R, beliefs are not retrieved by ACT-R's retrieval buffer, but by functions in LISP, the language underlying ACT-R. As such, all of the agent's beliefs are always accessible, and the additional ACT-R property 'activation' is not used and thus not specified.

$Means\_of\_Capability\_has\_Properties(Belief, reasoning_{BOA},$
$(FP_{Belief}, SP_{Belief}, DP_{Belief})$

$FP_{Belief} = (\langle declarative, yes\rangle, \langle accessible, yes\rangle, \langle time, T\rangle, \langle source, S\rangle,$
$\langle certainty, C\rangle)$

$SP_{Belief} = (\langle representation, chunk\rangle)$

$DP_{Belief} = \emptyset$

The $\text{Processes}_{\text{reasoning}}$ operate on the beliefs that the agent holds. These processes are task-specific; for BOA, they constitute its procedural knowledge on how to reason about its beliefs using domain knowledge. In ACT-R, procedural knowledge is represented by production rules, which is also the case for BOA. ACT-R adds an utility value to its production rules, but this is not used within BOA.

$$Process\_of\_Capability\_has\_Properties(Rule, reasoning_{BOA}, (FP_{Rule}, SP_{Rule}, DP_{Rule})$$

$$FP_{Rule} = (\langle task\_specific, yes \rangle)$$

$$SP_{Rule} = (\langle representation, production\text{-}rule \rangle, \langle deterministic, true \rangle)$$

$$DP_{Rule} = (\langle executability, E \rangle)$$

The $\text{Control}_{\text{reasoning}}$ determines which rules operate on which beliefs. In ACT-R, the control of reasoning takes one clear constraint into account; a maximum of one production rule may fire at a time, which is therefore also the case for BOA.

To determine which process may execute, the antecedent of a production rule is matched with the contents of ACT-R's buffers to determine the rule's executability. In principle, it is possible that the antecedents of multiple rules match, in which case ACT-R uses a conflict-resolution mechanism to decide which rule may fire. This entails that ACT-R's reasoning control is a form of decision-making. However, BOA was implemented in such a way that maximally one production rule can be activated at a time, which means that the conflict-resolution mechanism is never activated.

$$Control\_of\_Capability\_has\_Properties(C, reasoning_{BOA}, (FP_C, SP_C, DP_C)$$

$$FP_C = (\langle task\_specific, no \rangle, \langle considers\_executability, yes \rangle)$$

$$SP_C = (\langle representation, Lisp \rangle, \langle type\_of\_loop, single \rangle)$$

$$DP_C = (\langle number\_of\_active\_reasoning\_processes, 1 \rangle)$$

By choosing ACT-R for BOA's implementation some of BOA's reasoning capability properties were fixed. For example, because ACT-R only allows one production rule to fire at a time, BOA's reasoning control has as dynamic property that the number of active reasoning processes is always 1.

The specific reasoning capability variant presented here is an example of a *horizontal* variant, as the specified properties hold for any reasoning task agent BOA performs.

Decision-Making

Part of the tactical picture compilation task consists of employing an available helicopter to gather additional visual information about radar contacts. To do this in a correct way

one important decision has to be made: where to send the helicopter to. The specific vertical decision-making capability that enables this is elaborated on here.

The $\text{Means}_{\text{decision-making}}$ are alternatives from which one or several have to be chosen. For BOA, these alternatives are the current positions of the contacts on the radar screen: to one of these locations the helicopter should be sent.

$$Means\_of\_Capability\_has\_Properties(Positions, decision\text{-}making_{Heli}, (FP_{Pos}, SP_{Pos}, DP_{Pos}))$$

$$FP_{Pos} = (\langle declarativeness, true\rangle, \langle accessibility, true\rangle, \langle time, current\rangle, \langle urgency, U\rangle, \langle informativeness, I\rangle, \langle resource\_economics, R\rangle)$$

$$SP_{Pos} = (\langle representation, chunk\rangle)$$

$$DP_{Pos} = (\langle evaluation\_score, Relevancy\rangle)$$

Three $\text{Processes}_{\text{decision-making}}$ are identified within any decision-making capability. Also for this variant it holds that these processes do not have any required dynamic properties.

$$DP_{DO} = DP_{FO} = DP_{SO} = \emptyset$$

The first process, *Determination-of-Options*, determines which positions are eligible to send the helicopter to, which are positions of contacts that have not been visually identified before and not been identified as neutral or friendly.

$$FP_{DO} = (\langle task\_specific, yes\rangle, \langle considers\_constraint, yes\rangle)$$

$$SP_{DO} = (\langle representation, production\text{-}rule\rangle, \langle takes\_into\_account\_constraint, not\_identified\_visually\rangle, \langle takes\_into\_account\_constraint, no\_neutral\_id\rangle, \langle takes\_into\_account\_constraint, no\_friendly\_id\rangle)$$

The second process, *Evaluation-of-Options*, takes three aspects into account to calculate an evaluation score that represents how relevant it is to go to that location, namely a location's *urgency*, *informativeness*, and *resource economics*. The *urgency* of a location represents the time that is left to take precautionary actions in case the contact turns out to be hostile. Its *informativeness* is based on the number of contacts that can be identified visually from that location, as well as on their expected identity. The distance of a location towards the helicopter determines its *resource economics*.

$$FP_{EO} = (\langle task\_specific, yes\rangle, \langle considers\_aspect, yes\rangle)$$

$$SP_{EO} = (\langle representation, production\text{-}rule\rangle, \langle evaluation\_type, non\_injective\rangle, \langle takes\_into\_account\_aspect, urgency\rangle, \langle takes\_into\_account\_aspect, informativeness\rangle, \langle takes\_into\_account\_aspect, resourceeconomics\rangle)$$

The third process, *Selection-of-Options*, makes the actual selection and takes as constraint into account that no other location may have a higher score than the one selected.

$$FP_{SO} = (\langle task\_specific, yes\rangle, \langle considers\_constraint, yes\rangle, \langle considers\_evaluation\_score, yes\rangle)$$
$$SP_{SO} = (\langle representation, production\text{-}rule\rangle, \langle takes\_into\_account\_constraint, highest\_relevancy\rangle)$$

The $\text{Control}_{\text{decision-making}}$ simply executes the three processes in a single loop in the fixed order, by which it is decided where to send the helicopter to.

This specific decision-making capability variant presented here is an example of a *vertical* variant, as the specified properties do not hold for all, but only for a specific decision-making task of the agent.

6.2.6 Discussion and Conclusion

In this paper we presented a new approach for defining cognitive agent capabilities: by specifying the required properties for the capability entity types means, processes and control. With this approach it is possible to specify *generic* capabilities, i.e., processes that are considered horizontal modules by Fodor (1983), as well as *specific* variants, independent of whether these constitute a vertical or a horizontal module.

By means of examples we have shown how CaDeF can be applied to describe both conceptual and implemented capabilities. However, the reasoning of BOA is not very complex. In the future we will apply CaDeF's generic reasoning capability definition to other agents to evaluate whether it is also suitable to capture more complex reasoning variants. The treated examples do show that the framework offers room to vary the level of abstraction between entities, properties and thus capabilities. Entities can range from simple items to complex structures embedding processes and control themselves. This feature of the framework explicitly acknowledges the nature of agents and models of cognition where levels of description are not fixed and where capabilities come in many variants.

So, although CaDeF structures the description of cognitive agents through mandatory entities for describing capabilities, it provides enough flexibility to describe all types and variants of capabilities. Our aim is to start using these descriptions to catalog (components of) agents that possess these capabilities. Such a catalog is desired because it enables an agent designer to draw upon a pool of existing and implemented capabilities while designing a new agent. When this is possible, the development costs of new cognitive agent models are reduced, which in turn will lead to a more cost-effective construction of training simulations that embed cognitive agents.

In future work, the effectiveness of this approach will be further evaluated. In addition, its possible role in selecting an appropriate cognitive architecture for the development of an agent will be examined. Because different architectures provide more or less support for certain properties, our framework might aid to identify which architecture is most suited for a certain cognitive agent design.

Acknowledgments

This research has been supported by the research program 'Cognitive Modeling' (V524), funded by the Netherlands Defense Organization.

Chapter 7

Feedback System

7.1 Introduction

In the previous chapters we have focused on modeling components of cognitive agents and on the development of a method to describe cognitive capabilities. These efforts were conducted to develop cognitive agents that can generate human-like behavior in a simulated environment. However, we are also interested in using agents as instructors, e.g., to provide feedback to the student.

In this chapter we develop a training system based on software agents, tailored to the provision of feedback on task performance in dynamic, open, complex tasks such as situational assessment. Previously, we determined that for these types of tasks it is desired to provide feedback at the level of cognition, e.g., on the appropriateness of the followed strategy (Section 2.4.2).

To enable the generation of cognitive feedback to a student's task behavior, it is required to diagnose his or her task performance, for which two main techniques have emerged: *model-tracing* and *constraint-based modeling*, see Section 2.4.2. Feedback systems that embed model-tracing focus on inferring the (erroneous) *process* by which a student arrived at a solution, and base their feedback on this. In contrast, feedback systems based on constraint-based modeling base their feedback solely on the *final-state* the student arrived at, independent of the process that led him or her there.

In order to create a robust feedback system, we decided to combine variants of both techniques in one system. The system will first try to generate feedback at the level of diagnosed cognitive processes. However, it is not always possible to diagnose the cognitive process (e.g., due to multiple possible explanations) in which case the system can fall back on the generation of model-based feedback.

The developed feedback-generating training system is based on multiple software agents. It embeds an *expert* agent that performs the task according to expert standard, and a series of *deficient* agents that perform the task in a variety of wrong (biased) ways. In addition, the system incorporates a *student-behavior* agent that monitors the task behavior of the student. Last, we developed a *feedback-generating* agent (FeGA), which diagnoses the student's task performance and provides the feedback.

Chapter Overview

This chapter embeds one paper (Section 7.2) which focuses on the development of a multi-agent-based training system for dynamic, open, complex tasks. In this paper we mainly elaborate on the development of the Feedback Generating Agent (FeGA). FeGA evaluates and diagnoses the task performance of a student by retrieving information from the training environment and the other agents. Subsequently, FeGA provides feedback to this performance, which it can do either after a single, or a series of trials. We evaluate the capability of FeGA to diagnose task performance by letting it evaluate and diagnose software agents representing possible students.

Research Paper

7.2 FeGA: a Feedback-Generating Agent

Abstract

In current simulation-based training of knowledge-intensive tasks, human instructors are needed to evaluate a student's task performance. This paper reports a study into the development of a multi-agent-based training system that evaluates student behavior at the result-level (quality of performance) and at the process-level (appropriateness of taken approach). The system uses expert and error modeling as well as plan recognition to evaluate and diagnose the student behavior. Furthermore, it keeps track of this behavior over time and generates feedback on the student's task performance after either one trial or a series of trials. Exploratory results suggest that the system can correctly diagnose the behavior of students.

This section is published as:
Heuvelink, A., and Mioch, T. FeGA: A cognitive Feedback Generating Agent. In *Proceedings of the Seventh IEEE/WIC/ACM International Conference on Intelligent Agent Technology* (IAT 2008), p 567-572, IEEE Computer Society Press. December 9 - 12 2008, Sydney, Australia.

7.2.1 Introduction

A common way to teach a task is by training students in its execution. An important factor in task training is the generation of feedback on the student's task performance (Bosch and Riemersma, 2004). Usually instructors who possess expert task knowledge as well as didactic knowledge monitor the training, evaluate the student's performance and progress, and provide feedback. When the task environment is dangerous and uncertain, e.g., as faced by firemen and the military, task training often takes place in simulated environments. These environments guarantee the safety of the students and facilitate training because they are perfectly controllable.

Despite these benefits, students are still dependent for their training on the availability of instructors. It would be beneficial if students could train anytime, anywhere, and by themselves. However, in such a situation it is important to ensure that the training is conducted properly and no wrong behavior is learned. For this, the training environment should be able to monitor, evaluate, and diagnose the student's performance and provide feedback.

Many intelligent tutoring systems (ITS) have been developed that possess such feedback-generation methods, for overviews see (Polson and Richardson, 1988; VanLehn, 2006). However, most of these systems concern the training of (procedural) tasks in well-structured and small domains. The challenge is to generate feedback on task performance in open, complex domains for which not only the final result, but especially the process leading to this result is of importance.

This paper focuses on the development of a feedback-generation method that is capable of generating feedback on student performance for open tasks.

7.2.2 Types of Feedback

In the context of simulation-based training systems, three types of feedback can be distinguished (Mioch et al., 2007):

- **Result-based feedback**, also named minimal feedback (VanLehn, 2006), is based solely on the result of the task behavior of the student. Feedback is generated by comparing this result with the correct result, which is often hard-coded.
- **Model-based feedback** is not only based on the result of the student's behavior, but also on contextual knowledge of the simulation environment and explicit task knowledge. Feedback is generated by reasoning about the result of the student and why it was good or false, for which it uses an expert model and the task circumstances.
- **Cognition-based feedback** is also based on the student's result, the context of the task and explicit task knowledge, but additionally takes a student model into account

that tracks his or her behavior. Using this extra knowledge, feedback can be generated not only on the final result of the student's behavior, e.g., the selected action, but also on the process, e.g., how he or she selected this action.

Cognition-based and model-based feedback are also referred to as **error-specific feedback** (VanLehn, 2006). The two major approaches for generating error-specific feedback are *constraint-based modeling* and *model-tracing* (Kodaganallur et al., 2005). Both techniques have successfully been applied in ITS. However, most of these ITS concern the training of tasks in well-defined, small domains, for which it is relatively easy to deduce both types of error-specific feedback. ITS that have been developed for training tasks in open domains generally limit themselves to model-based feedback (Stottler and Vinkavich, 2000).

Generating error-specific feedback for open tasks is hard because often no single correct behavior exists. Additionally, the quality of the task performance depends for a great part on the student's reasoning process and not so much on the final result. Last, often a multitude of possible error-explanations exist from which it is hard to choose (Menzel, 2006).

Instructors commonly deal with these difficulties by simply asking the students to explain why they behaved the way they did. However, this is not a feasible strategy for an automated feedback system, since it is hard to implement a system that is capable of understanding a student's explanation of his or her behavior. In addition, instructors are known to try to deduce the cognitive strategies of students by paying attention to other behavioral factors related to the task, e.g., by monitoring where the student looks at (Bosch and Riemersma, 2004).

7.2.3 Training Open Tasks

The military is concerned with training open, complex tasks like decision-making in dynamic and uncertain conditions. A well-known paradigm for laboratory studies into learning such tasks is multiple cue probability learning (MCPL) (Brehmer, 1980), also called multidimensional functional learning (MFL) (Hoffman et al., 1981). Two important elements of MCPL/MFL tasks are a criterion that the student must learn to predict, and cues representing the information from which the student has to make a prediction. The student's goal is to learn how the criterion values relate to the cue values.

A military example of such a task is threat assessment, in which the threat of an object has to be judged based on a few factors, i.e., its attributes. To research the training of this task, a simulation environment was developed in GameMaker (Overmars, 2009). The simulation environment consists of a 2D map of an environment on which fifteen

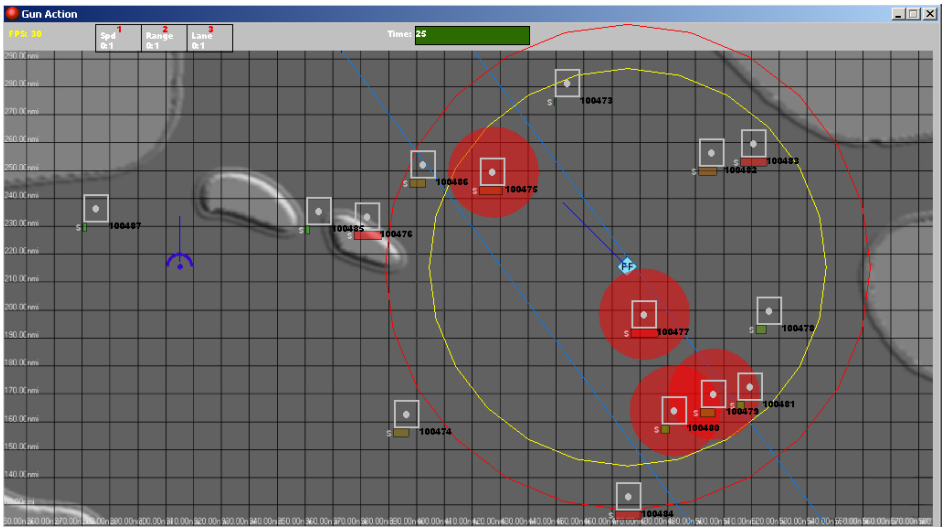


Figure 7.1: Screenshot of the simulation environment for training threat assessment

surface radar contacts are plotted, see figure 7.1. The own force is depicted by a small blue diamond (FF), the other known contacts as gray squares. On the map a sea lane is plotted, which denotes a common travel route for merchants from harbor to harbor. In this environment a student is trained to assess the threat of surface radar contacts on the basis of their speed, position relative to the own force, and position relative to sea lanes.

To adequately judge the threat levels of the contacts, the student needs to assess the value and importance of the various factors separately, relative to the other factors, and in combination. For example, the student needs to judge the importance of speed, whether speed is more important than position with respect to sea lanes when the contact is nearby, and what constitutes a higher threat: a contact that sails fast, outside a sea lane but does that far away, or a contact that sails slowly in the sea lane but does so in close proximity.

In a training trial the student has 30 seconds to assess the values of the several factors, after which he or she should select the contact judged to be most threatening. However, values of factors can not be assessed directly but have to be requested through **task-support buttons**. After pressing a button its information will be visible for a short time period. Figure 7.1 shows all the available information at the same time, as if all the buttons are pressed simultaneously. Button 1 shows the various *speeds* of the contacts (s-bar below the contacts), button 2 shows two *distance* ranges with tactical meaning (yellow and red circle) and button 3 shows the contacts that follow the *sea lane* (red discs). The latter means that not only the contact's position lies within the sea lane, but that its course is also following the sea lane.

7.2.4 Feedback-Generation Method

For training open tasks it is beneficial to have a feedback-generation method that can easily incorporate new findings and can be tailored to various types of students. This is ensured by implementing it as a multi-agent-based training system. Notice that the current study focuses on determining the appropriate content for feedback, and not on how and when this feedback is presented to the student (VanLehn, 2006).

Feedback Based on an Expert Model

For the generation of feedback, first, the result of the student's behavior is considered. The contact that is chosen by the student as most threatening is compared with the best choice given the circumstances, i.e., with the choice of a task expert. This choice is generated by an **expert agent**. In the current study it is assumed that one expert solution exists. However, for open tasks are, in general, multiple expert solutions possible. The proposed system can accommodate multiple good solutions by adding more expert agents.

When the student's choice equals the choice of the expert, it is determined that the feedback should be positive. When the choice does not cohere with that of the expert, other feedback has to be generated. Feedback generation is aimed at helping the student to learn the task, which can be achieved by making task knowledge explicit. Model-based feedback, introduced in Section 7.2.2, elaborates on why the student's choice is not the best choice using expert task knowledge. For the threat assessment task this knowledge entails knowing about the relevant factors and the relationship of their values to the criterion value, i.e., the threat level. In the current task three factors play a role:

- **Factor 1) Sea Lane**: In how far a contact is suspicious due to whether or not it follows a sea lane, denoted by a binary value. Contacts that do not follow a sea lane are more suspicious than those who do.
- **Factor 2) Speed**: In how far a contact is suspicious due to its speed; the faster, the more suspicious it is.
- **Factor 3) Distance**: In how far a contact's position makes it threatening, i.e. lies within the dangerous zone determined by the shooting range of the expected enemy.

The latter two factors are represented by a real value between 0 and 1, see figure 7.2. The three factors carry different weights in the determination of a contact's threat level. The following equation displays the relationship between a contact's threat value and its values for the three factors.

$$threat = (.2 * sea\_lane) + (.3 * speed) + (.5 * distance)$$

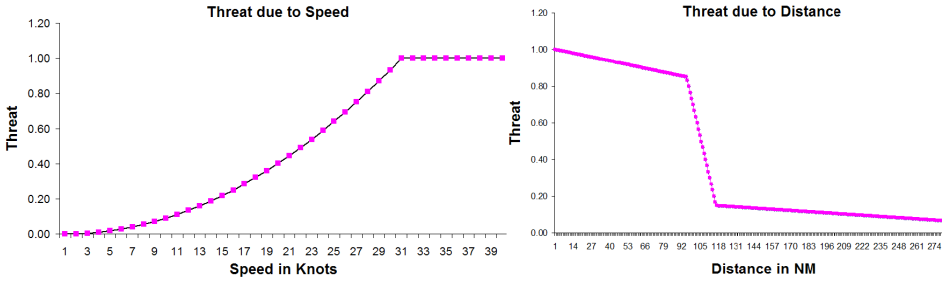


Figure 7.2: Threat due to a contact's speed (left) and due to its distance (right)

When the student has chosen a different contact than the expert, model-based feedback can be given on why that was a wrong choice. This feedback is generated by reasoning about how substantial the differences between the two choices are. What 'substantially different' entails is determined separately for each factor by a task expert. When the two choices do not differ substantially in any of the factors, it is deduced that the student's choice is also a valid option.

When the choices do differ substantially on a certain task aspect, that aspect is relevant for feedback. For example, when expert choice X lies within the dangerous zone but student choice S does not, it is deduced that the two choices differ in the factor *distance*. Since the expert choice has a substantial higher threat value due to the distance factor, it is further deduced that the student gives insufficient weight to that factor. This approach generates feedback of a similar nature and in a similar way as constraint-based modeling.

Feedback Based on Error Models

For this task it is easy to deduce model-based feedback. However, this type of feedback is limited in that it just considers the student's response, and only gives feedback on the level of task knowledge. Cognition-based feedback on the other hand also takes the reasoning process of the student into account and promises better learning results (Mioch et al., 2007).

The common way to generate cognition-based feedback is by deviation or error modeling (VanLehn, 2006), a technique also used by model-tracing. Typical errors for the current task are, e.g., the selection of the nearest or the fastest contact, thus giving excessive weight to a single factor and insufficient weights to the others. The rationale behind modeling errors is that when the student's behavior corresponds to a particular error, it can be inferred that that error probably has been made. To trace whether a student's response originates from a particular deficiency, it should be known to which choices the

deficiencies lead. These choices are generated by multiple **deficient agents**, with every agent incorporating a deficiency. The set of deficient agents can easily be altered, and therefore also the types and amount of errors the feedback-generation method is sensitive to.

Unlike model-based feedback, it is not always straight-forward to generate cognition-based feedback. Feedback cannot be deduced directly when the student's result does not match any of the results of the deficient agents, or when multiple deficient agents' results match the student's result.

In section 7.2.2, it was mentioned that instructors also use other behavioral aspects to deduce the reasoning process of the students, like where they look at. By extending the training environment with task-support buttons this process is made partially observable to the system. Since values of factors need to be requested, it can be deduced *which* factors the student attends to. Furthermore, because those values are only shortly visible, the amount of requests gives information on *how much* a student attends to certain factors.

By administering the task to experts it is determined what, on average, the required task-support button use is. When a student clearly under- or overuses a button, this is an indication of insufficient or excessive weighing of that factor. Assessing a student's reasoning process from his or her button use can be considered a form of plan recognition (VanLehn, 2006) and is done by the **student-behavior agent**.

Feedback Based on a Student Model

Two ways in which feedback can be generated on the behavior of a student after a trial have been discussed. However, neither of them can deduce after a single trail how well the student performs the task on average. To increase the reliability of the feedback on the general level of task performance, it is required that the student's behavior is tracked over multiple trials. For this a **student-performance model** is introduced, which consists of two layers. The lower layer keeps track of the task performance with respect to the various factors; the upper layer of overall task performance.

The lower layer keeps track of whether the student gives *insufficient*, *good* or *excessive* weight to a factor. This is represented by a value denoting the membership of this factor to the fuzzy sets denoting these weights. When it is assessed that the student gives an incorrect weight to a certain factor, the values of the fuzzy sets are updated with a step size of 0.2. When the student shows correct behavior, the fuzzy sets are updated with a step size of 0.05. This ensures that a single false behavior is not compensated by a single correct behavior; after false behavior the student should show correct behavior over multiple runs before the model deduces that a correct weight is given to a certain factor.

When the lower level deduces that all the weights given by the student are correct, the upper level deduces that the student has learned the task. The upper layer consists of membership values for the student's performance for the three fuzzy sets *unknown*, *known*, and *learned*. When the lower level deduces that a factor is given correct weight, this is transferred in an update of the known or learned set. Otherwise, the unknown set is updated. Over a series of trials the assessed correct and wrong aspects of the student behavior accumulate in the layers of the student-performance model. This leads to an increasing certainty considering the overall quality of the student's task performance.

Feedback Generation Agent (FeGA)

The last agent required by the multi-agent-based training system is the **feedback agent** that actually controls the generation of feedback after the student has finished the task.

For this, FeGA retrieves the result of the **student** and the **expert** and **deficient agents**. When the choice of the student does not correspond to any of the agent's choices it is not correct, but moreover, no cognition-based feedback can be generated. In such a case FeGA falls back on generating model-based feedback for which it uses task knowledge and information concerning the contacts and the environment.

When the choice of the student corresponds to one or more agents, it is checked for which agents this holds. In case the expert agent is amongst them, FeGA concludes that the student has performed the task well, and gives feedback accordingly. When the choice of the student matches that of a single deficient agent, it is concluded that the error originates from the deficiency represented by that agent. This results in cognition-based feedback based on that deficiency.

When the student's choice matches that of multiple deficient agents, FeGA will reason further over which of these deficient agents also matches the student's reasoning process. For this it first retrieves the assessment of whether the student paid good, insufficient or excessive attention to the various factors, formed by the **student-behavior agent**.

When the interpretation of the attention given by the student to the factors corresponds to the reasoning process of a deficient agent, this agent's deficiencies form the basis of feedback. When no deficient agent's reasoning process matches that of the student, it is checked whether all the deficient agents have the same deficiency for a particular factor, i.e., give the same false (insufficient or excessive) weight to it. If this is true, it can be assumed that that deficiency caused them to choose the wrong contact, and feedback is given concerning the wrong use of this single factor. When this is not true FeGA falls back on model-based feedback.

Feedback generated by FeGA is transferred to its **student-performance model**. The

Name	Sea lane	Speed	Distance
Expert	0.2	0.3	0.5
Student 1	0	0.4	0.6
Student 2	0.3	0	0.7
Student 3	0.4	0.6	0
Student 4	1	0	0
Student 5	0	1	0
Student 6	0	0	1
Student 7	0	0.5	0.5
Student 8	0.5	0	0.5
Student 9	0.5	0.5	0
Student 10	0.1	0.6	0.3
Student 11	0.3	0.6	0.1
Student 12	0.6	0.3	0.1

Table 7.1: Student weights given to the factors

generated feedback can be transferred directly to the student, or it can be decided to give feedback after multiple trials, using the knowledge accumulated in the student-performance model.

7.2.5 Evaluation

FeGA can be evaluated on two levels: whether it generates appropriate feedback for each *single* trial and whether its diagnosis about the student's *overall* task performance is correct. It was decided to evaluate FeGA by letting it evaluate and diagnose software agents representing possible students. This has the advantage that all aspects of the student's behavior are known, which is not the case when 'real' students are deployed for the evaluation. Another advantage is that the agents can be designed in such a way that many different ways of reasoning are covered.

It is assumed that the student agents have knowledge of the three relevant factors, but do not know the correct relation of their values to the threat value of a contact. In total 12 agents are modeled. See table 7.1 for the different weights the various student agents give to each factor.

The reasoning of students 1-6 corresponds to the reasoning of the modeled deficient agents. Students 7-9 exclude a factor all together, and weigh the other two factors equally. Students 10-12 are used to cover an even greater variety of ways to weight the factors. The light gray cells denote that the student agent ascribes insufficient (I) weight to that factor compared to the expert agent, dark gray cells that it ascribes excessive (E) weight, and white that it ascribes a good (G) weight.

Name	**Sea lane**			**Speed**			**Distance**		
weight:	*I*	*G*	*E*	*I*	*G*	*E*	*I*	*G*	*E*
Student 1	1	0	0	0	0	1	0	0	1
Student 2	0	0	1	1	0	0	0	0	1
Student 3	0	0	1	0	0	1	1	0	0
Student 4	0	0	1	1	0	0	1	0	0
Student 5	1	0	0	0	0	1	1	0	0
Student 6	1	0	0	1	0	0	0	0	1
Student 7	1	0	0	0	0	1	.2	.8	0
Student 8	0	0	1	1	0	0	0	.05	.95
Student 9	0	0	1	0	0	1	1	0	0
Student 10	1	0	0	0	0	1	.2	.8	0
Student 11	0	.05	.95	0	.05	.95	.95	.05	0
Student 12	0	.85	.15	0	1	0	.95	.05	0

Table 7.2: Final student-performance models

For evaluating FeGA 15 scenarios are generated, each containing 15 contacts that display a great variety in behavior. The behavior that the student agents show during task execution (i.e., the button-use to gather information about the contacts) corresponds to its deficient reasoning process, measured in relation to the average button-use of experts.

Results

Table 7.2 shows the student-performance models formed by FeGA for every student agent over the 15 trials.

It was found that students 1-6 choose at almost every trial a different contact than the expert agent. In the few cases that the student agent chooses the same contact as the expert, positive feedback is generated. When the choice does not match the choice of the expert, it usually matches the choice of the corresponding deficient agent. Even if there is a match with several deficient agents, the 'correct' one can be chosen as the student agent behaves according to his deficiencies. This means that for these trials valid cognition-based feedback is returned on the basis of the reasoning mistake of the matching deficient agent. Although sometimes positive feedback is generated, the accumulated values of the student-performance model over 15 trials perfectly correspond to, and thus correctly diagnose, the deficiencies of the student agents.

The deficient weighing of the factors by students 7-12 does not correspond to the weighing of any of the deficient agents. As a consequence, it is often not possible to diagnose a single deficient agent as representative for the student agent's behavior. The student agents' choices sometimes match the choice of the expert agent, the choice of

several other deficient agents, or not with any of the other agents' choices. Feedback after a trial is thus respectively sometimes positive, based on deficient agents whose deficiencies are not necessarily equal to those of the student agent, or model-based. As a result, the diagnosis of FeGA after a single trial is not very reliable. However, it was found that after 15 trials the accumulated values of the student-performance models correspond fairly well to the actual deficiencies of the student agents.

Only twice, for student 7 and 8, an accumulated value of the student-performance model does not correspond to the actual deficiency of the student agent, denoted in table 7.2 by light gray cells. Figure 7.3 shows the student-performance model of student 7 who weighs the factor sea lane *insufficiently* (I), the factor speed *excessively* (E), and the factor distance *well* (G). In none of the 15 trials the student chooses the same contact as the expert, as otherwise all the performance values would have come closer to zero at the same time. Further, FeGA does not diagnose after each trial that the student weighs the factor speed excessively. This is because several times the student's choice matches the choice of several deficient agents, none of which has the exact same deficiencies as the student. As a consequence, the student's behavior during the task performance (i.e., the button-pressing behavior) does not correspond to any of the deficient agents. This means that an agreement in deficiency between the matching deficient agents is returned as feedback, or model-based feedback is generated.

Fortunately, the accumulation of the feedback generated after each trial over 15 trials enables FeGA to correctly diagnose the student's excessive weighing of the factor speed. On the other hand, FeGA fails to diagnose the correct weighing of the factor distance by the student. The reason for this is that often the student's choice matches the choice of an agent that is deficient in the factor distance. However, FeGA is not very certain about its diagnosis of the factor distance, which is represented by the fluctuating value of the student-performance model for that factor.

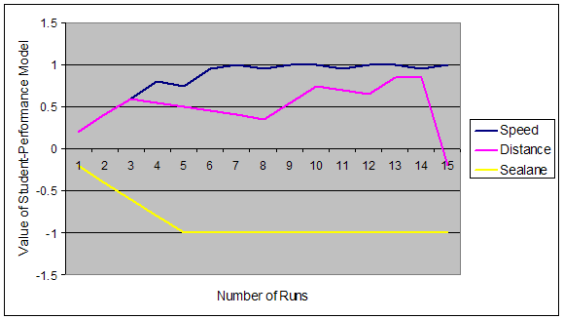


Figure 7.3: Student-7-Performance model

7.2.6 Discussion and Conclusion

We developed and implemented a feedback system based on multiple agents that together diagnose student performance and can generate feedback after a single trial or a series of trials. In certain aspects the developed feedback-generation method resembles constraint-based modeling and error modeling. Our work differs in that the feedback system incorporates both approaches as well as a student-behavior agent, which aids in dissolving possible ambiguities in the assessment of the student's task performance. For the latter it is required that the interaction of the student with the simulation environment is extended in such a way that extra knowledge concerning the student's behavior during task execution can be extracted.

Although the current evaluation with student agents yielded positive results, it is not yet certain whether equally positive results will be achieved with real students. The main reason for this is that for implementing the student agents certain simplifications were made. For example, it is assumed that the student agents have fixed deficiencies in their weighing of the important factors to assess which contact has the highest threat. Since the student agents do not learn, they will never adapt their deficiencies, even when provided with correct feedback. Real students are likely to learn and will therefore be more dynamic in their reasoning process. This will lead to a higher fluctuation in the values of the student-performance model and thus to a lower certainty of the accumulated feedback. However, this seems to be quite realistic: if the student does not have a consistent deficiency, no certain diagnosis about this deficiency can be given by the feedback agent.

Another issue is that currently positive feedback is returned when the choice of the student matches the choice of the expert, even when that choice also matches the choice of deficient agents. In such a situation it might be better to reason one step further, and check with which agent the student's behavior during the task execution matches best. When that behavior corresponds to the deficiencies of the deficient agent, it can be decided to return feedback that not simply mentions that the result was correct, but that also mentions the assessed reasoning deficiencies.

Because of the simplifications made, the developed feedback-generation method has to be further evaluated. A real indication of the validity of the feedback-generation method is when students are able to learn the task correctly based on the feedback that is generated. Therefore, in the next evaluation phase, it will be tested whether students that receive the feedback generated by FeGA learn quicker than students that receive simple result-based feedback. Subsequently, the generality of the developed feedback-generation method can be evaluated, e.g., by implementing it for a different open task with other relevant factors.

Acknowledgments

This research has been supported by the research program 'Cognitive Modeling' (V524), funded by the Netherlands Defense Organization. The authors like to thank Willem van Doesburg and Karel van den Bosch for useful discussions, Rob Verkuylen for the development of the simulated task environment, and Tijmen Muller for his help with Soar.

Chapter 8

Conclusion

The presented study was motivated by the gap between current scenario-based simulator training for which many people are required, and the desired situation in which students can train at any time by themselves. In order to narrow this gap, we developed methods and techniques for the modeling of intelligent agents that can perform roles in simulator training. It is likely that in the future such agents replace the humans currently required for training; by this our research contributes to future independent training.

We started this research with three research questions (Section 1.2.2). To answer these questions we developed in this study 1) components for cognitive agents that enable them to display rational as well as biased behavior, 2) a framework to describe cognitive capabilities, and 3) a feedback generating system based on multiple agents that diagnoses task performance. Our major research effort has been in the first area, so in developing *content* of cognitive agents. Our second research effort, in the *development* of cognitive agents, and third, in the *application* of cognitive agents, are at a more initial stage.

In the following sections we discuss for each research question the opportunities and limitations of the developed methods and techniques, as well as additional research possibilities. In the last section we reflect on the relevance of our work, and identify the need for future research.

8.1 Modeling Human-Like Behavior

The first research question of this study was: 'How can a cognitive agent display human-like behavior with a varying degree of biasedness?' Existing methods and techniques are commonly developed to bring about either rational or biased behavior, and not both types. We developed mechanisms that enable an agent to display rational as well as, to

a larger or smaller extent, biased behavior. For this, we extended the scope of Artificial Intelligence to more human-like behaviors inspired by Cognitive Science.

We investigated our first research question in the context of a typical military task: situational assessment. We started with modeling a rational, expert agent for this task. To achieve an assessment of a complex situation, an agent must be able to deal with factual, but also with uncertain information. In addition, it must be able to integrate information received from different sources, and relate new information to information previously gathered. These things are self-evident for humans, since most real-world tasks require these capabilities.

Unfortunately, reasoning about uncertain information received over time and from different sources is as yet not supported by integrated architectures. In general, they do not require information to be stored with a degree of belief, a time stamp, or source label, and therefore also do not offer support for the handling of these values. Usually, architectures do support that their basic knowledge entities can embed strings and numerical values, and often also offer operators that can compare them for equality and ordering. However, there exists a large gap between such generic processes and *cognitive* processes operating on these values.

A reason for the architectures stemming from Artificial Intelligence to not support the explicit reasoning over uncertain beliefs from different sources over time might be that these architectures are usually used to model agents for tasks that are better structured than the open, dynamic, complex tasks investigated here. Architectures stemming from Cognitive Science might not support these processes because they are usually used to model tasks at a smaller scale, and with more detail for the cognitive validity of the underlying processes, than the current research task.

We do not claim that the methods and techniques we developed are most suited to model human-like behavior for all types of tasks. For many tasks existing architectures possess all what is required for their modeling, and frequently more than we offer. However, they lack the required cognitive processes (e.g., reasoning about uncertainty or about time) for modeling dynamic, open, complex tasks like situational assessment. We hope that the mechanisms developed in this study can bridge the gap between the processes required to model complex tasks and those usually offered by integrated architectures.

8.1.1 Developed Cognitive Agent Capabilities

In order to perform its task in a simulated world, an agent must be able to reason about this world. We decided to model an agent's knowledge about what is going on by means of beliefs. For modeling a rational, expert agent in situational assessment we first had to develop methods and techniques that would enable agents to reason about uncertain

information received over time and from different sources. Next, for modeling behavior with a varying degree of biasedness, we had to determine which cognitive processes could become biased, as well as how and when this would happen.

In Chapter 3 we presented methods for an agent to transform observed information into beliefs, to integrate beliefs from different sources and/or times, and to reason about beliefs over time. These methods enable an agent to do this in a rational, or in a to a smaller or larger extent biased way. The implemented biases were selected by examining literature from Cognitive Science and by talking to task experts. The extent to which the biases influence the agent's behavior is determined by a parameter. This parameter was labeled 'stress' because of the context in which we want biases to emerge: under circumstances of stress and exhaustion. In order to model the agent's (biased) belief formation processes we proposed a belief notation that includes information about the time, source, and certainty of the information. At this stage, memory was modeled as an unlimited database of always available beliefs. This is not very human-like, and also turns out to be computationally undesirable.

In Chapter 4 we presented a more human-like memory model that distinguishes working/short-term memory from long-term memory. This memory model supports rational behavior due to the storage, maintenance, and perfect retrieval of beliefs, but also biased behavior, among others by forming abstract versions of beliefs, and by attaching an availability value to beliefs that enables biased retrieval. In particular, the memory model enables the inherent human memory aspect that sometimes specific details about an event can not be remembered anymore, but the event itself can. In addition, the model enables task-specific effects on memory, like the fact that beliefs required for the current task are readily available, and that not relevant beliefs are quickly forgotten.

In Chapter 5 we presented two control models that determine the behavior of the agent during task execution. The first model focuses on controlling the execution of the agent's cognitive processing components, and determines on-line whether rational or biased behavior emerges. The parameter that mediates between rational and biased behavior is labeled (cognitive) exhaustion. The mechanism that we developed to implement this dynamic exhaustion value was inspired by the idea of Hancock and Meshkati (1988) of 'tasks as stressors'. The essence of this idea is that humans get stressed when the tasks they have to execute lie above their cognitive capacity. This idea has, to our knowledge, not yet been implemented in cognitive agents to model stress. The second control model focuses on controlling the decision whether required information will be acquired by memory retrieval, or by sensing the world. For this model it is determined off-line which type of behavior emerges by the selection of one of the implemented, to a smaller of larger degree heuristic, task-strategies.

8.1.2 Points of Discussion

Belief Component

We decided to extend beliefs with a time stamp, source label and certainty value. Multiple aspects of this choice can be discussed. From an implementation point of view it is debatable whether the combination of these arguments with the belief term in one tuple was optimal, or whether it would have been better to have used multiple predicates. We selected the first option because of the increased transparency, although on multiple occasions it also demanded the specification of unused belief arguments.

Of another order is the question whether the added belief arguments are complete and sufficient. For the selected task it turned out they are. However, it is likely that for other tasks not all the arguments are relevant. In addition, for modeling other types of tasks and biases other arguments might be required, e.g., an emotion value.

Related to this point is the selection of the implemented biases, which are only a subset of all known biases. The reason to select the chosen biases was because they were known to influence situational assessment. The reason why we are satisfied with the implementation of this subset is that in order to create more human-like behavior it is, although desirable, not necessary to implement all possible emerging biases.

This point stresses the fact that we conduct *design* science: we 'devise artifacts to attain goals' and do not 'explain how and why things are'. This is also important when discussing the validity of the behavior generated by the agents embedding the developed belief component. We use the term validity to express the requirement that the behavior of the agents has observational fidelity, and not that this behavior is the result of cognitive valid processes. We validated this observational fidelity using experts (Section 3.3), but this validation study was unable to provide us with conclusive evidence on whether the belief framework is suited for modeling valid human-like behavior. The reason was that other factors (the agent's speed and task control) already impaired the face validity of the agent's behavior. Therefore, we cannot conclude with high certainty that the developed belief framework enables the modeling of human-like belief maintenance. However, the fact that we could use it to implement the situational assessment task and that it was successful in mimicking a specific case of biased human behavior, stems us hopeful.

The last discussion point follows from Section 3.1.2 in which we state that the belief arguments can be used to maintain beliefs in a variety of ways. Although we showed some ways in our research, we did not investigate how possible it is to implement other techniques using the same notation. It is likely that adjustments need to be made, e.g., for implementing rational source integration using Dempster's rule of combination, the certainty value should be transformed to a probability interval.

Memory Component

The structure of the proposed declarative memory model mimics, to our knowledge, none of the memories embedded in known integrated architectures. The memory model is inspired by the proposed belief representation and is unique in that it is purely episodic in nature. We propose that over time semantic memory can emerge from this episodic memory, due to aggregations that abstract from the episodic details of memories and combine their content. Most integrated architectures only embed semantic declarative memories, with Clarion and Soar as noticeable exceptions that embed episodic memory in addition to semantic memory. We acknowledge that Cognitive Science, starting from Tulving (1972), considers episodic and semantic memory to be two parallel, partially overlapping, but distinct information-processing systems. Nevertheless, semantic knowledge is not innate, but acquired during the lifetime of a person. In our view it is therefore defendable to model semantic memories as emerging from episodic memories, and to model them in one system.

The incentive to develop the memory model was to decrease the query time of the agent's belief base. We proposed to do this by deducing aggregated beliefs that are required for task execution, and by making them more available than the basic beliefs. Whether the memory model will indeed prevent the agent from slowing down awaits evaluation. An important point for this will be how frequently it is possible to have the beliefs that are required during task execution readily available. This is likely to vary from task to task.

Control Component

The developed agent reasoning control model determines on-line, by taking the task circumstances and its internal state into account, which reasoning components to execute. We developed this component to enable the modeling of agents that display rational behavior under normal task circumstances, but start showing biased behavior under stress, like humans do. In order to model this behavior the agent has to reason in a declarative way about many aspects, such as the possible actions and their relevancy. Most integrated cognitive architectures do not support such explicit, declarative, on-line reasoning over actions, let alone by taking into account an agent's internal state. We think that it is required for an agent to reason in a declarative way over its actions to execute the variety of behavior required for training simulations. One interesting question concerning the developed control for an agent's reasoning process, is its computational scalability. To determine which cognitive processing components are best executed it reasons over many aspects, which costs much processing time.

In the control model for information acquisition, the agent's behavior was determined by the selected task strategy. Besides the so-called 'rational' task strategy that reasoned about the costs and benefits of the various actions to choose between them, we implemented several 'heuristic' task strategies. The actions that could be selected by the latter were limited, and the heuristic strategies varied in the amount of information they retrieved in order to make a choice between the actions. The developed task model reasonably fitted the actions of humans executing the task. This is interesting, because the heuristic strategies were, although inspired by the participant's behaviors, derived by a meta-model on how to form heuristic strategies: by varying the number (and order) of retrieval actions humans are willing to take to come to a decision.

Of the participant's reaction times, only two out of four correlated with those of the model. The reaction times of the model that correlated the strongest with human reaction times was the model that followed a rational strategy. This, together with the fact that the behavior of the heuristic strategies appeared to be more restricted than the behavior actually shown by humans, makes it questionable whether the heuristic strategies truly capture the way humans operate. It might be more valid to adapt the rational strategy to fit various personalities. Currently, the strategies only consider the costs of the actions as they really are. However, it might be that people's personality influences how these costs are perceived, e.g., some people are very sensitive to making mistakes, so for them the costs of making a mistake should be weighted more. In addition, people's current internal state might influence these costs, e.g., when stressed or exhausted they might be slower in belief retrieval.

8.1.3 Additional Research

In this section we elaborate on ideas for future research that directly follow from the research presented in this study. At the end of this chapter, in Section 8.4, we elaborate on interesting future research directions of a more general order.

In this study we developed several agent components. A challenge for the future is to combine these components: the model that dynamically determines the agent's current exhaustion level can be used to make the fixed stress level embedded in the belief framework dynamic. In addition, the control model that decides whether rational or biased cognitive processing components execute, can be applied to the memory model to determine whether beliefs are perfectly retrieved, or 'quick and dirty'.

When the agent components are combined in one cognitive agent, it becomes more feasible to validate them by asking experts their opinion about the face validity of the agent's behavior. When the behavior is not judged to be observationally valid, it has to be established which of the embedded components causes that. Of course, it is also possible

that not these processes, but the embedded task knowledge causes it. This might be circumvented by implementing a task of which a well-established model exists (although it is not very likely to come across such a task-model for an open, dynamic, and complex task). An additional benefit of implementing a new task using the developed components is that this exercise will highlight how suited the developed mechanisms and techniques are for modeling other types of complex tasks.

By combining the components as proposed above, the biasedness of their processes will be tunable through one parameter. The advantage of this approach is that when a training instructor does not want agents to display biased behavior, or only wants them to do so, he or she can fix this parameter. This way the biasedness of all processes are controlled in one step. A disadvantage might be that this parameter combines a variety of cognitive states, like stress, cognitive exhaustion (workload), and fatigue. These states may have different effects, e.g., Harris et al. (2005) found that extended stress deteriorated performance, but not fatigue. On the other hand, Hancock and Desmond (2001) state that stress, workload, and fatigue are not distinct and separate phenomena, but actually only different facets of the same phenomenon: they are all reflections of the energetic state of an individual. For modeling observational valid behavior for training simulations, it is probably unnecessary to disentangle these states and their effects.

In future research, it would be useful to model more capabilities in such a way that they can display rational as well as biased behavior. An interesting question is how these biases will reinforce each other when the capabilities are combined. For example, at this moment an agent can display biased belief formation because of its trust in the source of the information. In the future, this trust in a specific source might also influence the decision to use, or to not use that source for acquiring information. When capabilities are combined and multiple biases operate at the same time, it is important to research how they reinforce each other and whether their effect needs to be proportionalized in order to keep the behavior believable.

It may be hard to determine the validity of the possible emerging interactions, because these have not been studied much. Teachman et al. (2007) investigated three different information processing biases to determine how they inter-relate, but found that the biases showed little relationship to one another. They state that the results of the few studies into multiple biases general suggest no significant correlations. For example, Lundh et al. (1999) found no correlation between memory and attention biases, and Lundh et al. (1997) found no relationship between measures of explicit memory and implicit memory biases. On the other hand, Hirsch et al. (2006) introduce the 'combined cognitive biases hypothesis' and propose that biases do not operate in isolation, but influence each other and interact.

A last point for future research is implementing the pattern-matching capability that enables experts to recognize a current situation as similar to a previous one (Klein, 1998). This capability inspired us to attach a time stamp to beliefs: time stamps enable the recognition of belief patterns in time. In addition, the memory model enables the formation of generic belief-pattern representations by abstracting from specific details of beliefs. These generic belief-pattern representations can be used to compare current specific beliefs with in order to determine a match. Therefore, we think the developed agent components are very suited for implementing pattern-matching.

8.2 Describing Agent Components

The second research question of this study was: 'How can cognitive agent capabilities be described?' We investigated this question because a uniform manner to describe capabilities, the typical content of cognitive agent components, can be used to label these components. These labels can, in turn, be used when searching for specific components to reuse when developing a new cognitive agent. So with this research, we hope to contribute to the cost-efficient modeling of cognitive agents.

8.2.1 Developed Capability Description Framework

We were motivated to develop a framework for describing capabilities of cognitive agents because in the literature there exists no consensus on what constitutes capabilities, or on how to describe them. We explained the wide variety of capability descriptions found to be a result of whether they are inspired by the agent's underlying cognitive theory, or by practical, task specific design choices. The first leads to definitions of so-called horizontal (agent-specific) capabilities such as 'reasoning', the second to definitions of so-called vertical (task-specific) capabilities such as 'multiply'.

The goal of the Capability Description Framework (CaDeF) is to be able to describe all these capability variants. For this, CaDeF 1) defines a method for describing (variants of) agent capabilities, and 2) provides definitions of generic agent capabilities. The method prescribes that capabilities are defined by specifying the *functional*, *system* and *dynamic* properties for three types of capability entities: *means*, *processes on means*, and *control of processes*. The combination of these entities make up the capability. The generic agent capability definitions can be used to describe horizontal as well as vertical variants of these generic capabilities. Specific capability variants are defined by specifying values for properties of the capabilities entities, or by specifying addition properties.

Our research is still at an initial stage: only two generic cognitive capabilities are

defined by studying the literature on cognitive agents and incorporating the common ideas shared. Also the evaluation of CaDeF is at a starting point. We evaluated CaDeF by using it to describe two variants of the defined definitions implemented in BOA. For these capabilities CaDeF turned out to be satisfactory: their definitions were enclosed by the generic definitions, and could be expressed by extending them.

8.2.2 Points of Discussion

We determined that each capability can be described using three types of entities. The main reason why we are confident that all capabilities can be described by defining properties for these entities on three different levels, is that these entities are defined by their functionality, and that each entity is allowed to embed multiple entities itself. However, it is a point of discussion whether we will be able to formally express the embedding of multiple entity types in another entity. Our major concern is whether it is possible to define generic rules for the inheritance of properties.

Another issue are the terms used to define properties. Because a wide variety of capability-variants exists, the terms used to specify properties should be free. An advantage of not predefining concepts is that this offers the freedom to integrate an arbitrary model (capability) that determines the required concept in a way suited for the task at hand. However, this freedom delivers a risk for the discoverability of components. For example, when an agent designer is interested in the capability to make decisions based on emotions, he or she might search for a decision-making capability with the functional property 'takes emotions into account'. When another agent designer has developed an agent that bases its decisions on its mood (which can be considered an emotional state) and has defined this agent's decision-making capability differently, e.g., as 'takes mood into account', this agent component will not be discovered. To deal with such situations it is useful to attach an ontology of cognitive terms to the search mechanism. An ontology enables the discovery of cognitive agent components based on description similarity. There exists a variety of ways in which ontologies can be formed and their content mapped to a search term, for an overview see Shvaiko and Euzenat (2005).

8.2.3 Additional Research

In future research we want to provide definitions for all the generic (cognitive) capabilities that cognitive agents embed. We suspect that only a limited amount of generic capability definitions is required, because of the freedom to add arbitrary properties to define specific variants, and the possibility to include one capability in another.

Each of these new, as the current, generic capability definitions need to evaluated

on their ability to enclose and capture all possible variants. For this, a wide variety of (implemented) cognitive agent models should be described using CaDeF. This effort will also inform us about the usability and effectiveness of CaDeF.

A last point we would like to investigate is the ability of CaDeF to describe the horizontal capabilities embedded in specific integrated architectures, and to use these descriptions to select an architecture for implementing a specific task model. When developing BOA and Boar, we encountered the difficulty of implementing an established task model in an established cognitive architecture. This undertaking made us aware of an important factor when developing agents: before selecting an architecture in which to implement an agent model, check whether the vertical capabilities the model requires are compatible with the horizontal capabilities of the architecture. In the future, CaDeF could deliver the means to check this compatibility.

8.3 Generating Cognitive Feedback

The third research question of this study was: 'How can an agent generate cognitive feedback to the behavior of a trainee?' Because our research concerns open, dynamic, complex tasks, the generation of feedback on the level of cognitive processes is hard.

8.3.1 Developed Feedback System

The approach we took for generating cognitive feedback was to build a robust system by combining several methods to diagnose performance. The feedback system is based on multiple agents. The *feedback-generating* agent (FeGA) provides feedback and diagnoses performance using the other agents. FeGA starts it diagnosis by comparing the performance of the student with that of *expert* and *deficient* agents (model-tracing). When multiple matches exist, it attempts to clear the confusion by comparing the strategies of these agents with the deduced strategy of the student (plan-recognition). The latter is provided to FeGA by the *student-behavior* agent that deduces this strategy from the interaction of the student with the training environment. To support this interaction, we proposed to extend the training environment with non-intrusive 'task-support-buttons'. When at this stage the student's performance is still undiagnosed, FeGA diagnoses it by comparing the student's result with the expert result (constraint-based modeling). The diagnosed performance of the student is stored in FeGA's *student-performance model*. This model denotes the student's performance for each relevant task aspect using fuzzy sets, which are updated after each training session. On the basis of its student-performance-model, FeGA generates cognitive feedback.

8.3.2 Points of Discussion

A principle question here is whether the feedback system will be suitable for training tasks that are more complex than the current one.

The task for which the feedback system is developed possesses important aspects for training real-world situational assessment tasks, namely a criterion (threat level) that the student must learn to predict based on cues (sea lane, speed, distance). However, the relation between the criterion values and cue values was straightforward in our study: only three cues had to be taken into account, their values were known, and these values were available at the same moment. In real-world situational assessment tasks there are often more cues relevant, and their values are usually uncertain and dynamic.

The first question that emerges for more complex tasks is whether we will be able to capture the required task knowledge in an expert agent model. The current expert and deficient agents are formed by simple models with limited performance. When the task becomes more complex, the performance of these models will as well, as will the number of deficient agents. Even when we are able to correctly capture (biased) task knowledge in agent models it is a question whether under such circumstances it is computationally feasible to 1) run all these agents on one machine, and 2) compare their performances on-line to the performance of the student. Another question is whether for more complex tasks and cues, the three fuzzy sets that the feedback system uses to denote the diagnosed performance of a student on a specific task aspect (cue) are suited. A last question is whether it is for all tasks possible to extend the training environment with facilities (e.g., task-support-buttons) that extract additional knowledge about the student's behavior during task execution, and, more importantly, whether it is also possible for more complex tasks to deduce cognitive strategies from that knowledge.

8.3.3 Additional Research

Foremost, it has to be validated whether the proposed feedback system is indeed capable of supporting the threat-assessment training. We evaluated FeGA's capability to diagnose performance by letting it evaluate and diagnose software agents representing possible students, which it did satisfactory. However, for the representation of the students certain simplifications were made, e.g., their reasoning process was considered to be static. A real indication of the validity of the feedback-generation method is when students are able to learn the task based on the feedback that is generated. Therefore, we are planning to test whether students that receive the feedback generated by FeGA learn the task quicker than students that receive simple result-based feedback.

When it is found that the feedback system is capable of supporting training for this

simplified situational assessment task, it can be researched whether it is also capable of training more complex tasks, i.e., with more uncertainty and higher dynamics. For the modeling of a feedback system for real complex, open, dynamic tasks, the cognitive agent components developed in this study might be useful. For the current task, the expert and deficient agents are formed by simple models which did not require, e.g., the reasoning over uncertain beliefs over time. However, for modeling more complex tasks this might be required, and than the developed components can be used. For this, first an expert agent has to be modeled that incorporates the developed capabilities whose biasedness can be tuned. Then, a set of agents can be formed, ranging from expert to deficient agents, by simply setting the stress/exhaustion parameter to a variety of values. However, it is a question whether the use of such agents to diagnose performance is scalable.

8.4 Future Research

In the previous sections we discussed how our study contributes to future independent training by developing methods and techniques for the 1) content, 2) development, as well as 3) application of cognitive agents. In this section we discuss future research concerning these aspects of cognitive agents.

8.4.1 Cognitive Agent Content

Sandercock (2004) identified several areas in which current computer generated forces consistently show weaknesses compared to human players: environment awareness, human variance, persistence, vengeance, anticipation, learning, and teaming. In the current study we have developed content for cognitive agents that, of these 7 weaknesses, will decrease the inability to show human variance and environment awareness the most.

In Section 1.4.4 we listed the processes a cognitive agent should be capable of in order to show human-like behavior for the situational assessment task. When we would have listed the capabilities required to show human behavior in all its facets, the list would have been considerably longer and include aspects such as empathy, communication, and learning. In order to support the independent training of all possible types of tasks, eventually all the known human capabilities need to be modeled, i.e., those listed by Gordon (2005) and Langley et al. (2006). This will support the development of agents that can validly represent human behavior in all its facets, for all types of tasks.

Situational assessment was selected as task because it is important within the military. It is a task that underlies many other military tasks, and it is potentially subject to a wide variety of cognitive biases, which makes it an important task to train. However, sit-

uational assessment is a task which can be executed by an individual (unlike many other military tasks), and this limited the requirements for the cognitive capabilities (content) of the agent. In future research the progression of human-like behavior in training simulations can be supported by modeling additional content for cognitive agents, e.g., content that enables cognitive agents to display varied, human-like behavior in team tasks. For team tasks it is required that agents are able to communicate; to reason about the beliefs, goals, and intentions of others (theory of mind reasoning); and to explain their own behavior. Because the cognitive agent components developed in this study incorporate mentalistic notions, and declaratively represent and explicitly reason about their reasoning rules, they are well suited to support these new capabilities. Other capabilities, e.g., learning, might require more research to implement using the developed components.

8.4.2 Cognitive Agent Development

Modeling cognitive agents following a component-based design approach is cost-effective when components are reused. To foster reusability, cognitive agent components should be placed in a repository that can be queried for useful components. Useful components are those that embed capabilities and properties required for modeling the task of the agent under construction. In this study we started the development of a Capability Description Framework (CaDeF) for capabilities of cognitive agents, whose descriptions can be used to label components so that they can be discovered for reuse.

Unfortunately, a capability description is not enough to determine whether a component can be reused. Tracz (1990) proposed that a component's description should contain: 1) the *concept* or abstraction the component represents; 2) the *content* of the component, or its implementation, and; 3) the *context* that component is defined under, or what is needed to complete the definition of a concept or content within a certain environment. CaDeF supports the description of the *concept* and *content* of a component by embedding functional and system properties of capabilities, respectively. However, CaDeF offers no support for denoting the *context* of a component. This aspect is interesting to investigate in future research, e.g., by specifying the relations between capabilities.

Future research into component-based development of cognitive agents can benefit from studying the modeling of agents within AI; these agents are frequently based on separate components and a coordination mechanism (see, e.g., Brazier et al., 2002; Bosse et al., 2007). In addition, much can be learned from the research fields of Component-Based Software Engineering (CBSE), and Web Services. For example, CBSE defines three common stages in the process of developing a system based on components, labeled component qualification, adaptation, and integration (Brown and Wallnau, 1996), and has developed methods and techniques for each of these stages. Similarly, Web Services are

concerned with the specification, discovery, and combination of specific services on the web, and have also developed many possibly useful methods and techniques.

In this study we only made a small step towards a structured, cost-efficient development methodology for cognitive agents. But recognizing that this is important aspect for their future application, and therefore keeping it in our mind when developing agents, is a gain compared to modeling agents in an ad-hoc fashion.

8.4.3 Cognitive Agent Applications

In this study we developed cognitive agents for displaying human-like behavior within a simulated environment, and for executing a training task in parallel with, but invisible to, a student. In the last case the agent's task performance was used for comparison with the performance of the student, which leads to a diagnosis of the student's task performance that is required for feedback generation.

The generation of feedback to a student's task performance is one of the functionalities defined by VanLehn (2006) for Intelligent Tutoring Systems (ITSs). Other functions of ITSs are constructing an individual-tailored curriculum, and answering questions about the exercises or domain in general. In this study we focused on using cognitive agents to generate feedback. Below we elaborate on how in future research cognitive agents might also aid the other two functionalities.

For constructing an individual-tailored curriculum, it is important to diagnose the student's task deficiencies, and to know which training scenarios will require the application of these deficiencies. When the latter is known, the curriculum can be tailored to the student by offering these scenarios. For open, dynamic, complex tasks it is hard to determine which specific task knowledge the scenario will draw upon. Possibly cognitive agents can aid to determine this, by performing the student's task in such a scenario.

For answering questions about the exercises or domain in general, cognitive agents can be useful, especially when they are extended with the capabilities to explain their own behavior and to perform theory-of-mind reasoning. The first capability enables the expert agent to provide generic answers to questions about the task knowledge it embeds, the second capability to tailor this explanation to the student. For example, when it is known that a student desires to know about one particular task aspect, the agent can especially elaborate on this aspect.

The methods and techniques for creating agents capable of showing varied human-like behavior could also be used in other applications than simulator training. They might be useful for developing and testing military doctrine for which nowadays, among others, Computer Generated Forces are used that do not always behave very human-like. In addition, they can be used for decision-support systems. Decision-support systems that

embed a rational task model can provide the system operator with suggestions on how to execute the task. Moreover, decision-support systems that embed knowledge about false task behavior might not only be capable of detecting that the operator's behavior is not rational, but also diagnose which mistake he or she made. When it is possible to diagnose false task behavior it is possible to extend the support suggestion with an explanation tailored to the operator, or to perform other types of actions that bring the diagnosed mistake to the operator's attention. The cognitive agent components might also be useful to implement agents for validating system design, e.g., to investigate which design decreases the emergence of biases the most, or for modeling more human-like companion agents.

In spite of these many directions for future research, we can envisage applications for cognitive models in the near future. This study focused on the development of cognitive models for complex tasks that would enable single persons to train by themselves, and this is still a challenge. However, the development of cognitive models for more procedural tasks is well possible, and could already enhance current simulation-based training. After all, any part of training in which a cognitive agent can replace a person is a gain, especially now the military faces a strong reduction in personnel. Similarly, cognitive models that perhaps cannot replace, but aid instructors so that multiple trainees can be trained at any one time are already possible. However, for training domains that face less pressure on available man-hours, a careful cost-benefit analyses should be made to determine whether the time and effort put in the development of cognitive models pays back by freeing up personnel.

8.5 Concluding Remark

The proverb 'time is money' dates back 2300 years: the favorite saying of the Greek philosopher Theophrastus ($\pm$ 300 B.C.) was 'time is the most valuable thing a man can spend', or more precisely: '$\pi o \lambda \upsilon \tau \epsilon \lambda \varepsilon \varsigma \ \alpha \nu \alpha \lambda \omega \mu \alpha \ \epsilon \iota \nu \alpha \iota \ \tau o \nu \ Χ \rho o \nu o \nu$'. Although many centuries later, also today much time and effort is spent on developing artifacts that in time will save us time. Indeed, more than that: no other century has yielded so many time-saving inventions as the previous one, and this trend is likely to continue in many years to come. A specific case in point is this study of which the objective is, in a nutshell, to decrease the man-hours required for training students in complex tasks.

We hope that this dissertation contributes to this time-saving trend by providing a clear story concerning the relevant questions, possible difficulties, as well as directions toward possible solutions to these questions and difficulties, when developing *Cognitive Models for Training Simulations*.

Appendix A

Overview of Software Packages

ACT-R	A cognitive architecture implementing a hybrid approach toward the modeling of cognition (Anderson and Lebiere, 1998). ACT-R constitutes a unified theory of cognition and is based on detailed findings concerning the functioning of human memory and of learning and problem-solving processes, see Section 2.3.2.
AI-implant	A commercial AI tool that can be used to simulate synthetic entities that make context specific decisions, and move in a realistic fashion within their environment. Its main use is simulating crowd behavior (Presagis, 2009).
COGNET	A framework for creating and exercising models of human operators engaged in primarily cognitive, as opposed to psychomotor tasks. COGNET's original use was the development of user models for intelligent interfaces, but it has also been used to model operators and opponents in simulators. The primary assumption underlying COGNET is that humans perform multiple tasks in parallel (Zachary et al., 1992, 1996).
CoJACK	A recent cognitive architecture that is used in simulation systems to underpin virtual actors, and created by adding a cognitive modeling layer on top of JACK, see below. CoJACK models the structural properties of the human cognitive system and constrains the models that can be implemented therein by only allowing the definition of models that fit within its structural boundaries. It models variation in human behavior by supporting behavior moderators such as stress, morale and fear, each implemented as an overlay (Norling and Ritter, 2004; Evertsz et al., 2008a).

CLARION	A cognitive architecture that explicitly distinguishes implicit from explicit processes and focuses on capturing the interaction between these two types of processes. It consist of four distinct subsystems: the action-centered subsystem, the non-action-centered subsystem, the motivational subsystem, and the meta-cognitive subsystem. Each of these subsystems has a dual representational structure, so incorporates implicit as well as explicit representations (Sun, 2002a).
D-Cog	Distributed Cognition (D-Cog) is a theoretical framework that takes a distributed, socio-technical system rather than an individual mind as its primary unit of analysis. This framework is explicitly cognitive in that it is concerned with how information is represented and how representations are transformed and propagated in the performance of tasks (Hutchins, 1995; Perry, 2003).
EPIC	A symbolic cognitive architecture developed for modeling human multiple-task performance, with as main focus the provision of a detailed account of human perceptual and motor operations. A parsimonious production system is used as cognitive processor, which is surrounded by separate sensory and motor processors. EPIC assumes that all capacity limitations are a result of limited structural resources, rather than a limited cognitive processor (Kieras and Meyer, 1997).
JACK	The JAVA Agent Compiler Kernel (JACK) is an extension to JAVA which implements a BDI architecture. It can be used to create a runnable JAVA program that instantiates a BDI agent. The agent's beliefs are represented with a database, its desires as events that can trigger plans, with these plans representing its intentions (Busetta et al., 2000).
Jason	An interpreter for an extended version of AgentSpeak (Rao, 1996), a BDI agent-oriented logic programming language. Jason is implemented in Java (Bordini et al., 2007) and forms a BDI agent. The agents belief state is the current state of the agent, which is a model of itself, its environment, and other agents. The agents desires are the states that the agent wants to bring about based on its external or internal stimuli, while its intentions are active, partially instantiated, plans that the agent adopts in an attempt to achieve its desires.
Java	An object-oriented, structured, imperative programming language, released in 1995 and originally developed by Sun Microsystems. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture.
Jess	A rule engine and scripting environment written in Java. Using Jess, Java software can be build that has the capacity to 'reason', using knowledge in the form of declarative rules (Friedman-Hill, 2003).

LeadsTo	A language and software environment developed to model and simulate dynamic processes in terms of both qualitative and quantitative concepts. Dynamic processes are modeled by specifying the temporal dependencies between state properties in successive states. The LeadsTo language is a declarative order-sorted temporal language, extended with quantitative means. The software environment performs simulations of LeadsTo specifications, generates simulation traces for further analysis, and constructs visual representations of traces. (Bosse et al., 2007).
PMFserv	An agent-based architecture with the flexibility to act as meta-level emotional arbitrator for others' cognitive architecture, or to provide a fully functional stand-alone system to simulate human decision-making. It provides a framework that permits examining the impacts of stress, culture, and emotion upon decision-making, and is mainly used to simulate crowd behavior (Silverman et al., 2006).
Prolog	A declarative, logic programming language, originally designed by Alain Colmerauer in 1972.
Soar	A symbolic cognitive architecture based on a production system and proposed by Newell (1990) as 'a candidate unified theory'. Soar implements goal-directed behavior as a search through a problem space, for which at each cycle first the current state is elaborated on, after which a decision is made which production rule (operator) may fire, see Section 2.3.2.
VBS2	Virtual Battlespace 2 is an interactive, three-dimensional training system providing a synthetic environment suitable for a wide range of military (or similar) training and experimentation purposes. VBS2 offers both virtual and constructive interfaces onto high-fidelity worlds and is used for mission rehearsal, tactical training and simulated combined arms exercises (Bohemia Interactive Australia, 2009).

Bibliography

Adelman, L. and Bresnick, T. (1992). Examining the effect of information sequence on patriot air defense officiers judgments. *Organizational Behavior and Human Decision Processes*, 53:204–228.

Adelman, L., Bresnick, T., Black, P., Marvin, F., and Sak, S. (1996). Research with patriot air defense officers: Examining information order effects. *Human Factors*, 38:250–261.

Adelman, L., Tolcott, M., and Bresnick, T. (1993). Examining the effect of information order on expert judgment. *Organizational Behavior and Human Decision Processes*, 56:348–369.

Alchourròn, C. E., Gärdenfors, P., and Makinson, D. (1985). On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530.

Aleven, V., Sewall, J., McLaren, B., and Koedinger, K. (2005). Rapid authoring of intelligent tutors for real-world and experimental use. In Kinshuk, Koper, R., Kommers, P., Kirschner, P., Sampson, D. G., and Didderen, W., editors, *Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006)*, pages 847–851. IEEE Computer Society.

Amant, R. S., Horton, T. E., and Ritter, F. E. (2007). Model-based evaluation of expert cell phone menu interaction. *ACM TRANSACTIONS ON COMPUTER HUMAN INTERACTION*, 14(1):Article 1 (24 pages).

Anderson, J. R. (1976). *Language, Memory and Thought*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Anderson, J. R. (1988). The expert module. In (Polson and Richardson, 1988), pages 21–54.

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Anderson, J. R., Bothell, D., Byrne, M., Douglass, S., Lebiere, C., and Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060.

Anderson, J. R., Bothell, D., Lebiere, C., and Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language*, 38:341–380.

Anderson, J. R., Boyle, C. F., Corbett, A. T., and Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42(1):7–49.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., and Pelletier, R. (1995). Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4(2):167–207.

Anderson, J. R. and Lebiere, C. (1998). *The Atomic Components of Thought*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Anderson, J. R. and Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological Science*, 2(6):396–408.

Anderson, N. H. (1981). *Foundations of Information Integration Theory*. Academic Press, Inc., New York, NY, USA.

Areces, C. and ten Cate, B. (2006). Hybrid logics. In Blackburn, P., Wolter, F., and van Benthem, J., editors, *Handbook of Modal Logics*. Elsevier.

Aristotle (350). *Nicomachean Ethics*. (translated by W.D. Ross).

Baddeley, A. D. (2000). The episodic buffer: a new component of working memory? *Trends in Cognitive Science*, 4:417–423.

Baddeley, A. D. and Hitch, G. J. (1974). Working memory. *Recent Advances in Learning and Motivation*, 8:647–667.

Bader, S. and Hitzler, P. (2005). Dimensions of neural-symbolic integration - a structured survey. In Artemov, S., Barringer, H., dAvila Garcez, A. S., Lamb, L. C., and Woods, J., editors, *We Will Show Them: Essays in Honour of Dov Gabbay*, volume 1, pages 167–194. College Publications.

Banks, S. B. and Stytz, M. R. (2003). Progress and prospects for the development of computer generated actors for military simulation: Part 2: Reasoning system architectures and human behavioral modeling. *Presence*, 12(4):422–436.

Barbuceanu, M. and Fox, M. S. (1996). The design of a coordination language for multi-agent systems. In *Proceedings of the ECAI'96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III*, volume 1193 of *LNAI*, pages 341–356. Springer-Verlag.

Baron, J. (2000). *Thinking and Deciding*. Cambridge University Press, Cambridge, UK.

Bayes, T. (1958 (originally published 1763)). An essay towards solving a problem in the doctrine of chances. *Biometrika*, 45(3-4):296–315.

Beer, R. D. (2000). Dynamical approaches to cognitive science. *Trends in Cognitive Sciences*, 4(3):91–99.

Beilock, S. L. and Decaro, M. S. (2007). From poor performance to success under stress: working memory, strategy selection, and mathematical problem solving under pressure. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 33(6):983–998.

Best, B. and Lebiere, C. (2006). Cognitive agents interacting in real and virtual worlds. In *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, pages 186–218. Cambridge University Press, New York, NY, USA.

Best, B., Lebiere, C., and Scarpinatto, K. C. (2002). Modeling synthetic opponents in MOUT training simulations using the ACT-R cognitive architecture. In *Proceedings of the Eleventh Conference on Behavior Representation In Modeling and Simulation (BRIMS 2002)*.

Bloch, I., Hunter, A., Appriou, A., Ayoun, A., Benferhat, S., Besnard, P., Cholvy, L., Cooke, R., Cuppens, F., Dubois, D., Fargier, H., Grabisch, M., Kruse, R., Lang, J., Moral, S., Prade, H., Saffiotti, A., Smets, P., and Sossai, C. (2001). Fusion: General concepts and characteristics. *International Journal of Intelligent Systems*, 16(10):1107–1134.

Bohemia Interactive Australia (2009). Virtual battle space 2. Retrieved from http://virtualbattlespace.vbs2.com/.

Bonsangue, M. M., Arbab, F., Bakker, J. W. D., Rutten, J. J. M. M., and Zavattaro, G. (2000). A transition system semantics for the control-driven coordination language manifold. *Theoretical Computer Science*, 240:3–47.

Bordini, R. H., Hbner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons-Interscience, New York, NY, USA.

Bosch, K. v. d. and Riemersma, J. (2004). Reflections on scenario-based training in tactical command. In Schiflett, S., Elliott, L., Salas, E., and Coovert, M., editors, *Scaled Worlds: Development, Validation and Applications*, pages 1–21. Aldershot, Ashgate.

Bosse, T. (2005). *Analysis of the Dynamics of Cognitive Processes*. PhD thesis, Vrije Universiteit Amsterdam.

Bosse, T., Jonker, C. M., van der Meij, L., and Treur, J. (2007). A language and environment for analysis of dynamics by simulation. *International Journal of Artificial Intelligence Tools*, 16(3):435–464.

Both, F. and Heuvelink, A. (2007). From a formal cognitive task model to an implemented ACT-R model. In *Proceedings of the 8th International Conference on Cognitive Modeling (ICCM 2007)*, pages 199–204. Psychology Press.

Boyd, J. (1996). The observe-orient-decide-act loop. Retrieved from http://www.d-n-i.net/richards/boyds_ooda_loop.ppt.

Bratman, M. E. (1987). *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, USA.

Brazier, F. M. T., Dunin-Keplicz, B., Treur, J., and Verbrugge, R. (2001). Modelling internal dynamic behaviour of BDI agents. In Gabbay, D. and Smets, P., editors, *Dynamics and Management of Reasoning Processes*, volume 6 of *Defeasible Reasoning and Uncertainty Management*

Systems, pages 339–361. Kluwer Academic Publishers.

Brazier, F. M. T., Jonker, C. M., and Treur, J. (2000). Compositional design and reuse of a generic agent model. *Applied Artificial Intelligence Journal*, 14(5):491–538.

Brazier, F. M. T., Jonker, C. M., and Treur, J. (2002). Principles of component-based design of intelligent agents. *Data Knowledge Engineering*, 41(1):1–27.

Brehmer, B. (1980). In one word: Not from experience. *Acta Psychologica*, 45(1-sup-3):223–241.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of [legacy, pre - 1988]*, 2(1):14–23.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–159.

Brown, A. W. and Wallnau, K. C. (1996). Engineering of component-based systems. In *Proceedings of the 2nd IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 1996)*, pages 414–422, Washington, DC, USA. IEEE Computer Society.

Browne, A. and Sun, R. (2001). Connectionist inference models. *Neural Networks*, 14(10):1331–1355.

Bryson, J. J. (2005). Modular representations of cognitive phenomena in AI, psychology and neuroscience. In Davis, D. N., editor, *Visions of Mind: Architectures for Cognition and Affect*, pages 66–89. Idea Group.

Busetta, P., Howden, N., Rönnquist, R., and Hodgson, A. (2000). Structuring BDI agents in functional clusters. In *6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL 1999)*, volume 1757 of *LNCS*, pages 277–289, London, UK. Springer-Verlag.

Byrne, M. D. and Kirlik, A. (2005). Using computational cognitive modeling to diagnose possible sources of aviation error. *International Journal of Aviation Psychology*, 15(2):135–155.

Byrne, M. D., Kirlik, A., and Fleetwood, M. D. (2008). An ACT-R approach to closing the loop on computational cognitive modeling: Describing the dynamics of interactive decision making and attention allocation. In Foyle, D. C. and Hooey, B. L., editors, *Human performance modeling in aviation*, pages 77–104. CRC Press, Boca Raton, FL, USA.

Byrne, R. M. J. and McEleney, A. (2000). Counterfactual thinking about actions and failures to act. *Journal of Experimental Psychology: Learning, Memory, and Cognition.*, 26(5):1318–1331.

Byrne, R. M. J. and Walsh, C. R. (2002). Contradictions and counterfactuals: Generating belief revisions in conditional inference. In Gray, W. and Schunn, C., editors, *Proceedings of the 24th Annual Conference of the Cognitive Science Society (CogSci 2002)*, pages 160–165, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Cadoli, M. and Donini, F. M. (1997). A survey on knowledge compilation. *AI Communications*,

10(3,4):137–150.

Carnegie Learning Inc. (2009). Cognitive tutor website. Retrieved from http://www.carnegielearning.com.

Castelfranchi, C. (1997). Representation and integration of multiple knowledge sources: issue and questions. In Cantoni, V., Ges, V. D., Setti, A., and Tegolo, D., editors, *Human & Machine Perception: Information Fusion*, pages 235–254. Plenum Press.

Castelfranchi, C. and Paglieri, F. (2007). The role of beliefs in goal dynamics: prolegomena to a constructive theory of intentions. *Synthese*, 155:237–263.

Chong, R. (1999). Towards a model of fear in soar. Retrieved from http://www.eecs.umich.edu/ soar/sitemaker/workshop/19/rchong-slides.pdf.

Ciancarini, P. (1996). Coordination models and languages as software integrators. *ACM Computings Surveys*, 28(2):300–302.

Cohen, P. R. and Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence*, 42(2–3):213–361.

Cohen, S., Evans, G. W., Stokols, D., and Krantz, D. S. (1986). *Behavior, Health, and Environmental Stress*. Plenum Press, New York, NY, USA.

Cowan, N. (2005). *Working memory capacity*. Psychology Press, New York, NY, USA.

Dam, B. J. v. and Arciszewski, H. F. R. (2002). Studie commandovoering do-2: Beeldvorming. Technical Report FEL-02-A242, TNO-FEL.

Davis, D. N. (2004). Why do anything? Emotion, affect and the fitness function underlying behaviour and thought. In *Proceedings of the AISB 2004 Symposium on Emotion, Cognition, and Affective Computing*, pages 21–32. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour.

Dempster, A. P. (1968). A generalization of bayesian inference. *Journal of the Royal Statistical Society, Series B*, 30:205–247.

Dickison, D. and Taatgen, N. A. (2007). ACT-R models of cognitive control in the abstract decision making task. In Lewis, R. L., Polk, T. A., and Laird, J. E., editors, *Proceedings of the Ninth International Conference on Cognitive Modeling (ICCM 2008)*, pages 79–84. Psychology Press.

Dieussaert, K., Schaeken, W., de Neys, W., and d'Ydewalle, G. (2000). Initial belief state as a predictor of belief revision. *Current Psychology of Cognition*, 19(3):277–286.

Diller, D. E., Ferguson, W., Leung, A. M., Benyo, B., and Foley, D. (2004). Behavior modeling in commercial games. In *Proceedings of the Thirteenth Conference on Behavior Representation In Modeling and Simulation (BRIMS 2004)*, Arlington, VI, USA. Simulation Interoperability Standards Organization.

Drosten, K. (1988). Translating algebraic specifications to Prolog programs: A comparative study. In *Proceedings of the International Workshop on Algebraic and Logic Programming*, volume 343 of *LNCS*, pages 137–146, London, UK. Springer-Verlag.

Dubois, D. and Prade, H. (2001). Possibility theory, probability theory and multiple-valued logics: A clarification. *Annals of Mathematics and Artificial Intelligence*, 32:35–66.

Elio, R. and Pelletier, F. J. (1997). Belief change as propositional update. *Cognitive Science*, 21(4):419–460.

Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems: Situation awareness. *Human Factors*, 37(1):32–64.

Engelmore, R. S. and Morgan, A. (1988). *Blackboard Systems*. Addison-Wesley, Reading, MA, USA.

Ernst, G. and Newell, A. (1969). *GPS: A Case Study in Generality and Problem Solving*. Academic Press, New York, NY, USA.

Evertsz, R., Ritter, F., Busetta, P., and Pedrotti, M. (2008a). Realistic behaviour variation in a BDI-based cognitive architecture. In *Proceedings of the Thirteenth annual Simulation Technology and Training conference (SimTecT 2008)*. Simulation Industry Association of Australia.

Evertsz, R., Ritter, F., Busetta, P., Pedrotti, M., and Bittner, J. (2008b). CoJACK - achieving principled behaviour variation in a moderated cognitive architecture. In *Proceedings of the Seventeenth Conference on Behavior Representation In Modeling and Simulation (BRIMS 2008)*, Arlington, VI, USA. Simulation Interoperability Standards Organization.

Fewell, M. P. and Hazen, M. G. (2005). Cognitive issues in modelling network-centric command and control. DSTO-RR 0293, Defence Science and Technology Organisation.

Fikes, R. and Nilsson, N. (1971). STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Fineberg, M. L. (1995). A comprehensive taxonomy of human behaviors for synthetic forces. Technical report, Institute for Defense Analyses, Alexandria, VA, USA.

Fodor, J. A. (1983). *The Modularity of Mind*. MIT Press, Cambridge, MA, USA.

Fogarty, W. (1988). Formal investigation into the circumstances surrounding the downing of Iran Air Flight 655 on 3 july 1988. Technical report, Department of Defense, Washington, DC, USA.

Fotta, M. E., Byrne, M. D., and Luther, M. S. (2005). Developing a human error modeling architecture (HEMA). In *Proceedings of Human-Computer International*, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Franklin, S. and Graesser, A. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. In Müller, J. P., Wooldridge, M. J., and Jennings, N. R., editors, *Proceedings of the*

ECAI'96 Workshop on Agent Theories, Architectures, and Languages: Intelligent Agents III (ATAL 1996), volume 1193 of *LNAI*, pages 21–36. Springer-Verlag.

Friedman-Hill, E. (2003). *Jess in Action. Java Rule-based Systems*. Manning Publications Co.

Galton, A. (2006). Operators vs arguments: the ins and outs of reification. *Synthese*, 150(3):415–441.

Gelder, T. v. (1995). What might cognition be, if not computation. *Journal of Philosophy*, 91:345–381.

Georgeff, M., Pell, B., Pollack, M., Tambe, M., and Wooldridge, M. (1999). The belief-desire-intention model of agency. In Müller, J., Singh, M. P., and Rao, A. S., editors, *Proceedings of the 5th International Workshop on Intelligent Agents: Agent Theories, Architectures, and Languages (ATAL-1998)*, volume 1555, pages 1–10, London, UK. Springer-Verlag.

Georgeff, M. P. and Lansky, A. L. (1987). Reactive reasoning and planning. In *Proceedings of the 6th National Conference on Artificial Intelligence*, pages 677–682, Menlo Park, California.

Gigerenzer, G., Todd, P. M., and Group, A. R. (1999). *Simple Heuristics that make us smart*. Oxford University Press.

Gilbert, D. T. and Malone, P. S. (1995). The correspondence bias. *Psychological Bulletin*, 117(1):21–38.

Gilbert, G. N. and Troitzsch, K. G. (1999). *Simulation for the Social Scientist*. Taylor & Francis, Inc., Bristol, PA, USA.

Gluck, K. A. and Pew, R. W., editors (2005). *Modeling human behavior with integrated cognitive architectures: comparison evaluation and validation.* Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Gomboc, D., Solomon, S., Core, M., Lane, H., and van Lent, M. (2005). Design recommendations to support automated explanation and tutoring. In *Proceedings of the Fourteenth Conference on Behavior Representation In Modeling and Simulation (BRIMS 2005)*, Arlington, VI, USA. Simulation Interoperability Standards Organization.

Gordon, A. S. (2005). Commonsense psychology and the functional requirements of cognitive models. In *Proceedings of the 2005 AAAI Workshop on Modular Construction of Human-like Intelligence*, Menlo Park, CA. AAAI Press.

Gratch, J. and Marsella, S. (2001). Tears and fears: Modeling emotions and emotional behaviors in synthetic agents. In *Proceedings of the 5th International Conference on Autonomous Agents*.

Gray, W. D. (2007a). Composition of integrated cognitive systems. In (Gray, 2007b), pages 3–12.

Gray, W. D., editor (2007b). *Integrated models of cognitive systems*. Oxford University Press, New York, NY, USA.

Gray, W. D. and Boehm-Davis, D. A. (2000). Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experiment Psychology: Applied*, 6:322–335.

Gray, W. D. and Fu, W.-T. (2004). Soft constraints in interactive behavior: The case of ignoring perfect knowledge in-the-world for imperfect knowledge in-the-head. *Cognitive Science*, 28:359–382.

Gray, W. D., Sims, C. R., Fu, W.-T., and Schoelles, M. J. (2006). The soft constraints hypothesis: A rational analysis approach to resource allocation for interactive behavior. *Psychological Review*, 113:461–482.

Groot, A. D. d. (1969). *Methodology: Foundations of Inference in Research in the Behavioral Sciences*. Mouton & Co, The Hague, the Netherlands.

Gunzelmann, G. and Anderson, J. (2001). An ACT-R model of the evolution of strategy use and problem difficulty. In *Proceedings of the Fourth International Conference on Cognitive Modeling (ICCM 2001)*, pages 109–114, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Gunzelmann, G., Gluck, K. A., Price, S., Dongen, H. P. A. V., and Dinges, D. F. (2007). Decreased arousal as a result of sleep deprivation. In (Gray, 2007b), pages 243–253.

Hancock, P. A. and Desmond, P. A., editors (2001). *Stress, Workload, and Fatigue*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Hancock, P. A. and Meshkati, N., editors (1988). *Human mental workload*. North-Holland, Amsterdam, the Netherlands.

Hancock, P. A. and Warm, J. S. (1989). A dynamic model of stress and sustained attention. *Human Factors*, 31(5):519–537.

Hanus, M. (1994). The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, 19:583–628.

Harris, W., Hancock, P., and Harris, S. (2005). Information processing changes following extended stress. *Military Psychology*, 17(2):115–128.

Hart, P. E., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Hasson, U. and Johnson-Laird, P. N. (2003). Why believability cannot explain belief revision. In Alterman, R. and Kirsh, D., editors, *Proceedings of the 25th Annual Conference of the Cognitive Science Society (CogSci 2003)*, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Hawkins, S. A. and Hastie, R. (1990). Hindsight: biased judgments of past events after the outcomes are known. *Psychological Bulletin*, 107(03):311–327.

Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321.

Hertwig, R. and Todd, P. (2003). More is not always better: The benefits of cognitive limits. In Hardman, D. and Macchi, L., editors, *Thinking: Psychological perspectives on reasoning, judgment, and decision making*, pages 213–231. John Wiley & Sons, Inc., Chichester, England.

Heuvelink, A. (2007). A belief framework for modeling cognitive agents. In *Proceedings of the 8th International Conference on Cognitive Modeling (ICCM 2007)*, pages 235–240. Psychology Press.

Heuvelink, A. and Both, F. (2007). BOA: A cognitive tactical picture compilation agent. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, pages 175–181. IEEE-CS Press.

Heuvelink, A., Klein, M. C. A., and Lambalgen, R. L. C. v. (2009a). Modeling human information acquisition strategies. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society (CogSci 2009)*. Cognitive Science Society. *In print.*

Heuvelink, A., Klein, M. C. A., and Treur, J. (2008a). An agent memory model enabling rational and biased reasoning. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2008)*, pages 193–199.

Heuvelink, A., Klein, M. C. A., and Treur, J. (2008b). A formal approach to belief aggregation. In *Proceedings of the 12th International Workshop on Cooperative Information Agents (CIA 2008)*, volume 5180 of *Lecture Notes of Artificial Intelligence*, pages 71–85. Springer-Verlag.

Heuvelink, A. and Mioch, T. (2008). FeGA: a feedback generation agent. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2008)*, pages 567–572.

Heuvelink, A., Mioch, T., and Doesburg, W. A. v. (2009b). CaDeF: Towards a method for describing cognitive agent capabilities. *Unpublished.*

Heuvelink, A. and Treur, J. (2008). Controlling biases in demanding tasks. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society (CogSci 2008)*, pages 1392–1397. Cognitive Science Society.

Hintzmann, D. (1986). "schema abstraction" in a multiple-trace memory model. *Psychological Review*, 93:411–428.

Hirsch, C., Clark, D., and Mathews, A. (2006). Imagery and interpretations in social phobia: support for the combined cognitive biases hypothesis. *Behavior Therapy*, 37(3):223–236.

Hoffman, P., Earle, T., and Slovic, P. (1981). Multidimensional functional learning (MFL) and some new conceptions of feedback. *Organizational Behavior and Human Decision Processes*, 27(1):75–102.

Hogarth, R. M. and Einhorn, H. J. (1992). Order effects in belief updating: The belief-adjustment model. *Cognitive Psychology*, 24:1–55.

Hollnagel, E. (1993). The phenotype of erroneous actions. *International Journal on Man-Machine Studies*, 39(1):1–32. kan.

Hulme, C., Roodenrys, S., Brown, G., and Mercer, R. (1995). The role of long-term memory mechanisms in memory span. *British Journal of Psychology*, 86:527–536.

Hutchins, E. (1995). How a cockpit remembers its speeds. *Cognitive Science*, 19:265–288.

James, W. (1890). *The principles of psychology (Vol. 1)*. Holt, New York, NY, USA.

Johnson, T. R., Wang, H., Zhang, J., and Wang, Y. (2002). A model of spatio-temporal coding of memory for multidimensional stimuli. In Gray, W. and Schunn, C., editors, *Proceedings of the 24th Annual Conference of the Cognitive Science Society (CogSci 2002)*, pages 506–511, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Johnston, W. A. and Heinz, S. P. (1978). Flexibility and capacity demands of attention. *Journal of Experimental Psychology: General*, 107:420–435.

Jones, R. M., Laird, J., Nielsen, P., Coulter, K., Kenny, P., and Koss, F. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine*, 20(1):27–41.

Jones, R. M., Lebiere, C., and Crossman, J. A. (2007). Comparing modeling idioms in ACT-R and Soar. In Lewis, R. L., Polk, T. A., and Laird, J. E., editors, *Proceedings of the Ninth International Conference on Cognitive Modeling (ICCM 2008)*. Psychology Press.

Juarez-Espinosa, O. and Gonzalez, C. (2004). Situation awareness of commanders: A cognitive model. In *Proceedings of the Thirteenth Conference on Behavior Representation In Modeling and Simulation (BRIMS 2004)*, Arlington, VI, USA. Simulation Interoperability Standards Organization.

Kahnemann, D. (1973). *Attention and Effort*. Prentice-Hall, Englewood Cliffs, NJ, USA.

Kahnemann, D., Slovic, P., and Tversky, A., editors (1982). *Judgement Under Uncertainty: Heuristics and Biases*. Cambridge University Press, Cambridge, UK.

Kieras, D. E. and Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12:391–48.

Klein, G. (1998). *Sources of Power*. MIT Press, Cambridge, MA, USA.

Kodaganallur, V., Weitz, R., and Rosenthal, D. (2005). A comparison of model-tracing and constraint-based intelligent tutoring paradigms. *International Journal of Artificial Intelligence in Education*, 15:117–144.

Kodaganallur, V., Weitz, R., and Rosenthal, D. (2006). An assessment of constraint-based tu-

tors: A response to Mitrovic and Ohlssons critique of "A comparison of model-tracing and constraint-based intelligent tutoring paradigms". *International Journal of Artificial Intelligence in Education*, 16:291–321.

Konieczny, S. and Pino Prez, R. (2002). Merging information under constraints: A logical framework. *Journal of Logic and Computation*, 12(5):773–808.

Laird, J. E., Congdon, C. B., and Coulter, K. J. (2006). The Soar users manual version 8.6.3. Available at http://ai.eecs.umich.edu/soar/sitemaker/docs/manuals/Soar8Manual.pdf.

Laird, J. E. and Duchi, J. (2000). Creating human-like synthetic characters with multiple skill levels: A case study using the Soar quakebot. In Freed, M., editor, *Proceedings of the 2000 AAAI Fall Symposium: Simulating Human Agents*, pages 54–58.

Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). SOAR: an architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64.

Langley, P., Laird, J. E., and Rogers, S. (2006). Cognitive architectures: Research issues and challenges. Technical report, Computational Learning Laboratory, CSLI, Stanford University, CA.

Lebiere, C., Anderson, J., and Reder, L. (1994). Error modeling in the ACT-R production system. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society (CogSci 1994)*, pages 555–559, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Lehman, J. F., Laird, J., and Rosenbloom, P. (2006). A gentle introduction to Soar, an architecture for human cognition: 2006 update.

Lewis, R. L. (2001). Cognitive theory, Soar. In Smelser, N. J. and Baltez, P. B., editors, *International Encylopedia of the Social and Behavioral Sciences*, pages 2178–2183. Pergamon (Elsevier Science), Amsterdam, the Netherlands.

Lucas, A. and Goss, S. (1999). The potential for intelligent software agents in defence simulation. In *Proceedings of the 1999 Conference on Information, Decision and Control (IDC 1999)*, pages 579–583.

Lundh, L., Czyzykow, S., and Öst., L. (1997). Explicit and implicit memory bias in panic disorder with agoraphobia. *Behaviour Research and Therapy*, 35(11):1003–1014.

Lundh, L., Wikstrom, J., and Westerlund, J. (1999). Preattentive bias for emotional information in panic disorder with agoraphobia. *Journal of Abnormal Psychology*, 108(2):222–232.

Maes, P. (1991). Situated agents can have goals. In Maes, P., editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 49–70. MIT press, London, UK.

Maes, P. (1994). Modeling adaptive autonomous agents. *Artificial Life*, 1(1-2):135–162.

March, S. T. and Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, 15:251–266.

Marinier, R. and Laird, J. (2007). Computational modeling of mood and feeling from emotion. In *Proceedings of the 29th Annual Conference of the Cognitive Science Society (CogSci 2007)*, pages 461–466.

Matthews, G. and Desmond, P. (2002). Task-induced fatigue states and simulated driving performance. *Journal of Experimental Psychology: Section A*, 55(2):659–686.

McBride, S., Merullo, D. J., Johnson, R., Banderet, L., and Robinson, R. (2007). Performance during a 3-hour simulated sentry duty task under varied work rates and secondary task demands. *Military Psychology*, 19(2):103–117.

McTear, M. (1993). User modeling for adaptive computer systems: A survey of recent developments. *Artificial Intelligence Review*, 7:157–184.

Menzel, W. (2006). Constraint-based modeling and ambiguity. *International Journal of AIED*, 16(1):29–63.

Mercier, H. and der Henst, J.-B. V. (2005). The source of beliefs in conflicting and non conflicting situations. In Bara, B., Barsalou, L., and Bucciarelli, M., editors, *Proceedings of the 27th Annual Conference of the Cognitive Science Society (CogSci 2005)*, pages 1495–1500, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psycological Review*, 63:81–97.

Minsky, M. (1992). Future of AI technology. *Toshiba Review*, 47(7).

Mioch, T., Harbers, M., Doesburg, W. A. v., and Bosch, K. v. d. (2007). Enhancing human understanding through intelligent explanations. In Bosse, T., Castelfranchi, C., Neerincx, M., Sadria, F., and Treur, J., editors, *Proceedings of the First International Workshop on Human Aspects in Ambient Intelligence*, pages 73–83, Darmstadt, Germany.

Mitrovic, A., Koedinger, K., and Martin, B. (2003). A comparative analysis of cognitive tutoring and constraint-based modelling. In Brusilovsky, P., Corbett, A., and de Rosis, F., editors, *Proceedings of the Ninth International Conference on User Modeling (UM 2003)*, volume 2702 of *Lecture Notes in Artificial Intelligence*, pages 313–322, London, UK. Springer-Verlag.

Mitrovic, A., Martin, B., and Suraweera, P. (2007). Intelligent tutors for all: The constraint-based approach. *IEEE Intelligent Systems*, 22(4):38–45.

Mitrovic, A. and Ohlsson, S. (2006). A critique of Kodaganallur, Weitz, and Rosenthal, "A comparison of model-tracing and constraint-based intelligent tutoring paradigms". *International Journal of Artificial Intelligence in Education*, 16:277–289.

Mitrovic, A., Suraweera, P., Martin, B., Zakharov, K., Milik, N., and Holland, J. (2006). Authoring

constraint-based tutors in ASPIRE. In Ikeda, M., Ashley, K., and Chan, T.-W., editors, *Proceedings of the eigth International Conference on Intelligent Tutoring Systems (ITS 2006)*, volume 4053 of *Lecture Notes in Computer Science*, pages 41–50, London, UK. Springer-Verlag.

Miyake, A. and Shah, P., editors (1999). *Models of working memory: Mechanisms of active maintenance and executive control*. Cambridge University Press, Cambridge, UK.

Morgan, G., Ritter, F. E., Stevenson, W., Schenck, I., and Cohen, M. A. (2005). dTank: An environment for architectural comparisons of competitive agents. In Allender, L. and Kelley, T., editors, *Proceedings of the Fourteenth Conference on Behavior Representation In Modeling and Simulation (BRIMS 2005)*, pages 133–140, Arlington, VI, USA. Simulation Interoperability Standards Organization.

Morrison, J. (2003). A review of computer-based human behavior representations and their relation to military simulations. IDA P-3845, Institute for Defense Analyses, Alexandria, VA, USA.

Moulin, B., Irandoust, H., Belanger, M., and Desbordes, G. (2002). Explanation and argumentation capabilities: Towards the creation of more persuasive agents. *Artificial Intelligence Review*, 17:169–222.

Muller, T. J., Heuvelink, A., and Both, F. (2008). Implementing a cognitive model in ACT-R and Soar: A comparison. In *Proceedings of the 6th International Workshop on From Agent Theory to Agent Implementation (AT2AI-6 2008) in conjunction with AAMAS 2008*.

Neerincx, M. A. (2007). Modelling cognitive and affective load for the design of human-machine collaboration. In *Proceedings of the Seventh international conference on Engineering Psychology and Cognitive Ergonomics (EPCE 2007), held as part of HCI International 2007*, volume 4562 of *Lecture Notes in Computer Science*, pages 568–574. Springer.

Newell, A. (1990). *Unified Theories of Cognition*. Harvard University Press, Cambridge, MA, USA.

Newell, A. and Simon, H. (1963). GPS: A program that simulates human thought. In Feigenbaum, E. A. and Feldman, J., editors, *Computers and Thought*, pages 279 – 293. McGraw-Hill, New York, NY, USA.

Newell, A. and Simon, H. (1976). Computer science as empirical enquiry: Symbols and search. *Communications of the Association for Computing Machinery*, 19(3):113–126.

Norling, E. and Ritter, F. (2004). A parameter set to support psychologically plausible variability in agent-based human modelling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 758–765, Washington, DC, USA. IEEE Computer Society.

Norling, E. J. (2004). Folk psychology for human modelling: Extending the BDI paradigm. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 202–209, Washington, DC, USA. IEEE Computer Society.

Norman, D. and Bobrow, D. (1975). On data-limited and resource-limited processes. *Cognitive Psychology*, 7:44–64.

Nuxoll, A. and Laird, J. E. (2004). A cognitive model of episodic memory integrated with a general cognitive architecture. In *Proceedings of the Sixth International Conference on Cognitive Modeling (ICCM 2004)*, pages 220–225.

Nuxoll, A. and Laird, J. E. (2007). Extending cognitive architecture with episodic memory. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2007)*, pages 1560–1564. AAAI Press.

Nuxoll, A., Laird, J. E., and James, M. R. (2004). Comprehensive working memory activation in Soar. In *Proceedings of the Sixth International Conference on Cognitive Modeling (ICCM 2004)*, pages 226–230.

Nwana, H. (1996). Software agents: an overview. *Knowledge Engineering Review*, 11(3):1–40.

Oser, R. L. (1999). A structured approach for scenario-based training. In *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting*, pages 1138–1142, Santa Monica, CA, USA.

Overmars, M. (2009). Designing games with game maker. Retrieved from http://www.yoyogames.com/gamemaker.

Padgham, L. and Lambrix, P. (2005). Formalisations of capabilities for BDI-agents. *Autonomous Agents and Multi-Agent Systems*, 10(3):249–271.

Paglieri, F. (2004). Data-oriented belief revision: Towards a unified theory of epistemic processing. In E. Onaindia, S. S., editor, *STAIRS 2004: 2nd Starting AI Researchers' Symposium*, pages 179–190. Amsterdam: IOS Press.

Parsons, S., editor (2001). *Qualitative Methods for Reasoning under Uncertainty*. The MIT Press, Cambridge, MA, USA.

Perrin, B. M., Barnett, B. J., and Walrath, L. C. (1993). Decision making bias in complex task environments. In *Proceedings of the Human Factors and Ergonomics Society 37th Annual Meeting*, pages 1117–1121, Santa Monica, CA, USA.

Perry, M. (2003). Distributed cognition. In Carroll, J., editor, *HCI Models, Theories, and Frameworks: Toward an Interdisciplinary Science*, pages 193–223. Morgan Kaufmann publishers Inc., San Mateo, CA, USA.

Pew, R. W. and Mavor, A. S., editors (1998). *Modeling human and organizational behavior: Application to military simulations.* National Academy Press, Washington, DC, USA.

Pohl, R. F., editor (2004). *Cognitive Illusions*. Psychology Press.

Polson, M. C. and Richardson, J. J., editors (1988). *Foundations of Intelligent Tutoring Systems*.

Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Posner, M. and Boies, S. (1971). Components of attention. *Psycological Review*, 78:391–408.

Presagis (2009). Ai-implant website. Retrieved from http://www.presagis.com/products/simulation/details/aiimplant/.

Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away (MAAMAW 1996)*, pages 42–55, London, UK. Springer-Verlag.

Rao, A. S. and Georgeff, M. P. (1991). Modeling agents within a BDI-architecture. In Fikes, R. and Sandewall, E., editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484, San Mateo, CA, USA. Morgan Kaufmann publishers Inc.

Rasmussen, J. (1983). Skills, rules, knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics*, 13:257–266.

Reason, J. (1988). Stress and cognitive failure. In Fisher, S. and Reason, J., editors, *Handbook of life stress, cognition and health*, pages 405–423. John Wiley & Sons, Inc., London, UK.

Reason, J. T. (1990). *Human Error*. Cambridge University Press, Cambridge, UK.

Ritter, F., Baxter, G., Jones, G., and Young, R. (2001). User interface evaluation: How cognitive models can help. In Carroll, J., editor, *Human-computer interaction in the new millennium*, pages 125–147. Addison-Wesley, Reading, MA, USA.

Ritter, F., Shadbolt, N., Elliman, D., Young, R., Gobet, F., and Baxter, G. (2003). Cognitive architectures: Research issues and challenges. Technical report, Defense Technical Information Center, Wright-Patterson Air Force Base, OH, USA.

Ritter, F. E., Kase, S., Bhandarkar, D., Lewis, B., and Cohen, M. (2007a). dTank updated: Steps towards exploring moderator-influenced behavior in a small synthetic environment. In *Proceedings of the Sixteenth Conference on Behavior Representation In Modeling and Simulation (BRIMS 2007)*, pages 51–60, Arlington, VI, USA. Simulation Interoperability Standards Organization.

Ritter, F. E., Reifers, A., Klein, A. C., and Schoelles, M. (2007b). Lessons from defining theories of stress. In (Gray, 2007b), pages 254–262.

Rollings, A. and Adams, E. (2003). *Andrew Rollings and Ernest Adams on Game Design*. New Riders Publishing.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach (2nd ed.)*. Prentice Hall, Upper Saddle River, NJ, USA.

Ryder, J. M., Santarelli, T., Scolaro, J., Hicinbothom, J., and Zachary, W. W. (2000). Comparison of cognitive model uses in intelligent training systems. In *Human Factors and Ergonomics Society Annual Meeting Proceedings*, volume 44, pages 374–377, Santa Monica, CA, USA. Human Factors and Ergonomics Society.

Ryder, J. M., Weiland, M. Z., Szczepkowski, M. A., and Zachary, W. W. (1998). Cognitive engineering of a new telephone operator workstation using COGNET. *International Journal of Industrial Ergonomics*, 22(6):417–429.

Sandercock, J. (2004). Lessons learned for construction of military simulations: A comparison of artificial intelligence to human-controlled agents. DSTO TR-1614, Defence Science and Technology Organisation Systems Sciences Laboratory, Adelaide, South Australia.

Schooler, L. J. and Hertwig, R. (2005). How forgetting aids heuristic inference. *Psychological Review*, 112(3):610–628.

Scott, B. (2002). The illusion of intelligence. In Rabin, S., editor, *AI Game Programming Wisdom*, pages 16–20. Charles River Media.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press.

Shahar, Y. (1997). A framework for knowledge-based temporal abstraction. *Artificial Intelligence*, 90(1-2):79–133.

Shields, S. (1983). Development of autonomic nervous system responsitivity in children: A review of the literature. *International Journal of Behavioral Development*, 6:291–319.

Shortliffe, E., editor (1976). *Computer-Based Medical Consultations: MYCIN*. Elsevier / North Holland, New York, NY, USA.

Shvaiko, P. and Euzenat, J. (2005). A survey of schema-based matching approaches. *Journal on Data Semantics IV*, 3730:146–171.

Silverman, B. G., Johns, M., Cornwell, J., and OBrien, K. (2006). Human behavior models for agents in simulators and games: Part I: Enabling science with PMFserv. *Presence*, 15(2):139–162.

Simon, H. (1967). *The Sciences of the Artificial*. MIT Press, Cambridge, MA, USA.

Simon, H. A. (1956). Rational choice and the structure of the environment. *Psychological Review*, 63:129–138.

Simon, H. A. (1997). Allen Newell: March 19, 1927 - July 19, 1992. In *Biographic Memoirs*, volume 71. National Academy of Sciences.

Sloman, A. (1997). What sort of architecture is required for a humanlike agent? In Wooldridge, M. and Rao, A., editors, *Foundations of Rational Agency*, pages 35–52. Kluwer Academic, New York, NY, USA.

Sloman, A., Chrisley, R., and Scheutz, M. (2005). The architectural basis of affective states and processes. In Fellous, J.-M. and Arbib, M., editors, *Who Needs Emotions?: The Brain Meets the Machine*, pages 203–244. Oxford University Press, New York, NY, USA.

Sloman, S. A. (1996). The empirical case for two systems of reasoning. *Psychological Bulletin*, 119:3–22.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–23.

Sripada, S. M. (1993). A temporal approach to belief revision in knowledge bases. In *Proceedings of the 9th Conference on Artificial Intelligence for Applications (CAIA 1993)*, pages 56–62, Orlando, FL, USA. IEEE Computer Society Press.

Stottler, R. and Vinkavich, M. (2000). Tactical action officer intelligent tutoring system (TAO ITS). In *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC 2000)*.

Strien, P. v. (1997). Towards a methodology of psychological practice: the regulative cycle. *Theory Psychology*, 7:683–700.

Stytz, M. R. and Banks, S. B. (2003a). Progress and prospects for the development of computer generated actors for military simulation: Part 1: Introduction and background. *Presence*, 12(3):311 325.

Stytz, M. R. and Banks, S. B. (2003b). Progress and prospects for the development of computer generated actors for military simulation: Part 3: The road ahead. *Presence*, 12(6):629–643.

Sun, R., editor (2002a). *Duality of the Mind: A Bottom-up Approach Toward Cognition*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Sun, R. (2002b). Hybrid systems and connectionist implementationalism. In Nadel, L., editor, *Encyclopedia of Cognitive Science*, volume 1, pages 697–703. Macmillan.

Teachman, B. A., Smith-Janika, S. B., and Saporito, J. (2007). Information processing biases and panic disorder: Relationships among cognitive and symptom measures. *Behaviour Research and Therapy*, 45(8):1791–1811.

Thagard, P. (1992). Adversarial problem solving: Modeling an oponent using explanatory coherence. *Cognitive Science*, 16(1):123–149.

Thagard, P. (2000). *Coherence in Thought and Action*. MIT Press, Cambridge, MA, USA.

Todd, P. M. and Gigerenzer, G. (2000). Precis of simple heuristics that make us smart. *Behavioral and Brain Sciences*, 23:727–780.

Tozour, P. (2002). The evolution of game AI. In Rabin, S., editor, *AI Game Programming Wisdom*, pages 3–15. Charles River Media.

Tracz, W. (1990). The 3 cons of software reuse. In *Proceedings of the 3rd Workshop on Software Reuse*.

Tulving, E. (1972). Episodic and semantic memory. In E.Tulving and Donaldson, W., editors, *Organization of memory*, pages 381–403. Academic Press, New York, NY, USA.

Tulving, E. (2002). Episodic memory: From mind to brain. *Annual Review of Psychology*, 53:1–25.

Tversky, A. and Kahneman, D. (1974). Judgment under uncertainty: Heuristics and biases. *Science*, 185:1124–1131.

Tversky, A. and Kahneman, D. (1982). Judgments of and by representativeness. In (Kahnemann et al., 1982), pages 84–98.

US-Army (2002). FM 17-96, the reconnaissance, surveillance, and target acquisition (RSTA) squadron of the brigade combat team. Technical report, U.S. Department of the Army, Washington, DC, USA.

Van Lent, M., McAlinden, R., Brobst, P., Silverman, B., O'Brien, K., and Cornwell, J. (2004). Enhancing the behavioral fidelity of synthetic entities with human behavior models. In *Proceedings of the Thirteenth Conference on Behavior Representation In Modeling and Simulation (BRIMS 2004)*, pages 1495–1500, Arlington, VI, USA. Simulation Interoperability Standards Organization.

VanLehn, K. (1988). Student modeling. In (Polson and Richardson, 1988), pages 55–78.

VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of AIED*, 16:227–265.

Walsh, C. R. and Sloman, S. A. (2004). Revising causal beliefs. In Forbus, K., Gentner, D., and Regier, T., editors, *COGSCI '04: Proceedings of the 26th Annual Conference of the Cognitive Science Society*, pages 1423–1427, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Wang, H. and Johnson, T. (1998). UEcho: A model of uncertainty management in human abductive reasoning. In Gernsbacher, M. A. and Derry, S. J., editors, *Proceedings of the 20th Annual Conference of the Cognitive Science Society (CogSci 1998)*, pages 1113–1118, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Wang, H., Zhang, J., and Johnson, T. (2000). Human belief revision and the order effect. In Gleitman, L. and Joshi, A., editors, *Proceedings of the 22nd Annual Conference of the Cognitive Science Society (CogSci 2000)*, pages 547–552, Mahwah, NJ, USA. Lawrence Erlbaum Associates, Inc.

Weiland, M., Campbell, G. E., Zachary, W., and Cannon-Bowers, J. A. (1998). Applications of cognitive models in a combat information center. In *Proceedings of 1998 Command and Control Research and Technology Symposium*, pages 770–782.

Wickens, C. D. and Flach, J. (1988). Information processing. In Wiener, E. L. and Nagel, D. C.,

editors, *Human Factors in Aviation*, pages 111–155. Academic Press, San Diego, CA, USA.

Wielemaker, J. (2003). An overview of the SWI-prolog programming environment. In *Proceedings of the 13th International Workshop on Logic Programming Environments*, pages 1–16.

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, 9(4):625–636.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152.

Wray, R. E., Laird, J. E., Nuxoll, A., and Jones, R. M. (2002). Intelligent opponents for virtual reality trainers. In *Proceedings of the Interservice/Industry Training, Simulation and Education Conference (I/ITSEC 2002)*.

Wray, R. E., Laird, J. E., Nuxoll, A., Stokes, D., and Kerfoot, A. (2004). Synthetic adversaries for urban combat training. In *Proceedings of Innovative Applications of Artificial Intelligence*. AAAI Press.

Zachary, W., J.Cannon-Bowers, Bilazarian, P., Krecker, D., Lardieri, P., and Burns, J. (1999). The advanced embedded training system (AETS): An intelligent embedded tutoring system for tactical team training. *International Journal of Artificial Intelligence in Education*, 10:257–277.

Zachary, W., Le Mentec, J.-C., and Ryder, J. (1996). Interface agents in complex systems. In Ntuen, C. and Park, E., editors, *Human Interaction with Complex Systems: Conceptual Principles and Design Practice*, pages 35–52. Kluwer Academic Publisher, Norwell, MA, USA.

Zachary, W., Ryder, J., and Hicinbothom, J. (1998). Cognitive task analysis and modeling of decision making in complex environments. In Cannon-Bowers, J. and Salas, E., editors, *Decision Making Under Stress: Implications for Training and Simulation*, pages 315–344. APA Press, Washington, DC, USA.

Zachary, W., Ryder, J., Ross, L., and Weiland, M. (1992). Intelligent human-computer interaction in real time, multi-tasking process control and monitoring systems. In Helander, M. and Nagamachi, M., editors, *Human Factors in Design for Manufacturability*, pages 377–402. Taylor and Francis, New York, NY, USA.

Zhang, J., Johnson, T., and Wang, H. (1998). Order effects and frequency learning in tactical decision making. *Thinking and Reasoning*, 4(2):123–145.

Zsambok, C. E. and Klein, G., editors (1997). *Naturalistic Decision Making*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Proefschrift Samenvatting

In dit proefschrift, getiteld “Cognitieve modellen voor trainingssimulaties”, hebben we onderzoek gedaan naar modellen van de menselijke cognitie met de achterliggende vraag hoe zulke cognitieve modellen kunnen bijdragen aan de training van personen in gesimuleerde omgevingen. In het bijzonder hebben we gekeken naar de vraag of cognitieve modellen gebruikt kunnen worden om de mensen te vervangen die nu vaak nodig zijn voor simulatietraining (instructeurs en rollenspelers).

In **hoofdstuk 1** wordt de motivatie voor dit onderzoek uitvoerig beschreven en verkennen we de ambitie om mensen in trainingssimulaties te vervangen door intelligente software, ook wel ‘agenten’ genoemd. Uit die verkenning zijn drie onderzoeksvragen voortgekomen. De eerste onderzoeksvraag is: *Hoe kan een cognitieve agent gedrag vertonen dat net zoals het gedrag van mensen variabel en foutgevoelig kan zijn?* Mensen gedragen zich namelijk lang niet altijd rationeel, maar laten zich - vaak onbewust - leiden door allerlei interne en externe omstandigheden. Als we de effecten van deze omstandigheden op gedrag goed kunnen representeren in een model, dan kunnen we dit model vervolgens gebruiken om agenten te bouwen die zich ‘menselijk’ gedragen. Het onderzoek naar deze eerste vraag vormt het grootste deel van dit proefschrift. De tweede onderzoeksvraag is: *Hoe kunnen capaciteiten van cognitieve agenten beschreven worden?* Deze vraag is relevant omdat voor het kosteneffectief ontwikkelen van cognitieve agenten het belangrijk is dat eerder ontwikkelde capaciteiten hergebruikt kunnen worden. De derde onderzoeksvraag heeft betrekking op een functie die normaliter een instructeur in een simulatietraining vervult: *Hoe kan een agent terugkoppeling geven op het taakresultaat én op de taakuitvoering van een student?*

We hebben deze drie vragen voornamelijk onderzocht binnen het domein van de Nederlandse marine. In de inleiding gaan we daarom ook kort in op het militaire domein, met name op het begrip *situational assessment*, dat situatiebeoordeling betekent en wat leidt tot *situational awareness*, oftewel situationeel bewustzijn. Situatiebeoordeling ligt ten grondslag aan vele militaire taken en wordt dan ook uitvoerig getraind.

In **hoofdstuk 2** geven we een overzicht van bestaand onderzoek dat gerelateerd is aan onze onderzoeksvragen. Vooral studies uit de cognitiewetenschappen en de kunstmatige intelligentie zijn voor dit proefschrift van belang. De cognitiewetenschap doet voornamelijk empirisch-experimenteel onderzoek naar het gedrag van mensen en ontwikkelt modellen op basis van die bevindingen. Dit verschilt van de modellen en instrumenten ontwikkeld binnen de kunstmatige intelligentie: deze zijn doorgaans slechts geïnspireerd door de menselijke cognitie.

In **hoofdstuk 3** onderzoeken we hoe we het menselijk vermogen om een situationeel bewustzijn op te bouwen zo kunnen modelleren dat een cognitieve agent dat ook kan. Mensen interpreteren een situatie in de wereld door gebruik te maken van allerlei typen informatie, zoals observaties, kennis-van-de-wereld, verwachtingen, enzovoort. Om dit soort informatie te representeren binnen een agent maken we gebruik van *beliefs*, dat letterlijk vertaald overtuigingen betekent. Het woord overtuiging geeft aan dat het geen feiten zijn waarmee geredeneerd wordt, maar interpretaties van feiten. Aan elk van deze overtuigingen verbinden we dan ook een zekerheidswaarde, bron en tijdslabel. Met behulp van deze extra informatie kunnen we de agent zo met zijn overtuigingen om laten gaan, dat die in meer of mindere mate beïnvloed kunnen worden door interne en externe omstandigheden. Met behulp van een historisch voorbeeld tonen we aan dat het met onze aanpak mogelijk is een typisch menselijke foutieve situatiebeoordeling na te bootsen. Vervolgens gebruiken we deze aanpak om een tactische beeldopbouwer van de marine te modelleren. Dit model hebben we geïmplementeerd in de cognitieve architectuur ACT-R. De resulterende agent kan een tactisch beeld opbouwen op zowel een rationele, alsook op een meer menselijke manier. Om te laten zien dat het door ons ontwikkelde model generiek is en niet gebonden is aan een specifieke architectuur, hebben we de agent geherimplementeerd in Soar, een andere cognitieve architectuur.

In **hoofdstuk 4** presenteren we onderzoek naar hoe een agent op een menselijke manier overtuigingen kan opslaan en herinneren. In hoofdstuk 3 legden we uit dat een agent met behulp van overtuigingen zich een beeld van de wereld kan vormen. Dit beeld is niet statisch, maar dynamisch, waarbij de ene overtuiging de andere kan overschrijven. Gedateerde overtuigingen blijven wel bewaard, zodat er geredeneerd kan worden over gebeurtenissen in de tijd (de vaarrichting van een schip kan bijvoorbeeld afgeleid worden uit twee overtuigingen over diens positie). Het opslaan van gedateerde overtuigingen heeft echter wel tot gevolg dat er een onhanteerbare grote hoeveelheid overtuigingen ontstaat, die bijvoorbeeld de in het vorige hoofdstuk geïntroduceerde beeldopbouwagent zienderogen deed vertragen. In dit hoofdstuk stellen we dan ook twee mechanismen voor om de hoeveelheid overtuigingen waarmee de agent kan redeneren hanteerbaar te houden. Het eerste mechanisme betreft de bewerking van specifieke overtuigingen tot

a) samengestelde overtuigingen (bijv. een overtuiging over de maximale snelheid van een schip uit een set van overtuigingen over de snelheid van dat schip) en b) tot geabstraheerde overtuigingen (bijv. een overtuiging dat een schip snel heeft gevaren, zonder precies te weten hoe snel of wanneer dat was). Het tweede mechanisme betreft de toekenning van een activatiewaarde aan overtuigingen. De activatiewaarde is afhankelijk van hoe vaak en hoe recent een overtuiging gebruikt is en bepaalt of de agent de overtuiging nog uit zijn geheugen kan oproepen of niet. Aan de hand van een voorbeeld demonstreren we de werking van het geheugenmodel. Hierbij focussen we ons op de mogelijkheid van het model om preciese gedetailleerde redeneringen te ondersteunen, als ook grovere, minder accurate gedachtestappen.

In **hoofdstuk 5** onderzoeken we hoe we kunnen modelleren dat het gedrag van cognitieve agenten, net zoals dat van mensen, in meer of mindere mate beïnvloed wordt door interne en externe omstandigheden. De eerder ontwikkelde componenten voor de overtuigingen en het geheugen van een cognitieve agent zijn in staat om dit gedrag te ondersteunen, maar ze zijn niet gebouwd om te bepalen welk gedrag optreedt. In dit hoofdstuk ontwikkelen we allereerst een component die dit kan bepalen. Uit de cognitiewetenschap is bekend dat het gedrag van mensen op een hoog niveau beïnvloed wordt door de doelen die iemand zichzelf stelt. Hoe iemand vervolgens uiting geeft aan deze doelen is afhankelijk van de omstandigheden, bijv. van de taaklast en de tijdsdruk. Om dit te modelleren hebben we twee mechanismen ontwikkeld. Het eerste mechanisme bestaat uit een deliberatiemodel waarmee de agent bepaalt - op basis van zijn doelen, de beschikbare cognitieve denkkracht en het stressniveau - welke cognitieve processen worden uitgevoerd. Het tweede mechanisme is een stressmodule die berekent hoeveel stress er optreedt als functie van de cognitieve taakbelasting. Met deze mechanismen tezamen kunnen we een dynamisch stressniveau creëren. Aan de hand van een voorbeeld laten we zien dat een agent onder hoge stress andere keuzes maakt (in het bijzonder cognitief 'goedkopere' redeneeracties kiest, die mogelijk tot een foutieve situatiebeoordeling leiden) dan wanneer hij geen of weinig stress ervaart.

Een groot deel van de menselijke cognitie bestaat uit het afleiden van informatie uit andere informatie. Als een agent een specifieke redeneerregel heeft gekozen om zijn doel te bereiken (bijv. het bepalen van de snelheid van een schip door deze af te leiden uit twee overtuigingen over diens positie met verschillende tijdslabels) dan is het voor het uitvoeren van zo'n regel nodig dat bepaalde informatie beschikbaar is (in dit geval de twee overtuigingen over de positie van het schip). De tweede component die we in dit hoofdstuk ontwikkelen heeft als functie om te bepalen hoe dit type informatie het beste beschikbaar gemaakt kan worden, namelijk a) door het te observeren in de wereld of b) door het op te halen uit het geheugen. Om te onderzoeken wat mensen doen, hebben

we een experiment opgezet met een taak waarin specifieke informatie nodig is die op verschillende manieren verkregen kon worden. Het experiment bestond uit twee condities: in de ene conditie waren de kosten van het opvragen van informatie in de wereld groter dan de kosten van het maken van fouten, in de andere conditie was het omgekeerde het geval. Op basis hiervan verwachtten we verschillend gedrag te zien, maar dit bleek niet zo te zijn. Naar aanleiding van dit resultaat hypothetiseren we dat mensen voor deze taak niet in staat zijn om door middel van een rationele kostenbaten analyse een keuze te maken tussen de mogelijke acties, maar dat mensen deze keuze bepalen door een andere, heuristische taakstrategie. Om dit te onderzoeken hebben we verschillende taakstrategieën gemodelleerd en hebben we een cognitieve agent geïmplementeerd die aan de hand van een bepaalde strategie de taak uitvoert. Om onze hypothese te toetsen hebben we de empirische gegevens van de mensen met de data van de agent vergeleken. Uit deze vergelijking bleek dat het gedrag van de mensen dankzij de ontwikkelde taakstrategieën redelijk door de agent nagebootst kon worden, maar dat er voor het maken van een compleet juist model nog verder onderzoek nodig is.

In **hoofdstuk 6** doen we onderzoek naar het kosteneffectief ontwikkelen van cognitieve agenten. Eén van de motivaties voor het onderzoek gepresenteerd in dit proefschrift is dat met cognitieve modellen mogelijk rollenspelers, die doorgaans nodig zijn voor simulatietraining, vervangen kunnen worden. Als we met cognitieve agenten de menselijke inzet kunnen verminderen, kan dit een aanzienlijke kostenbesparing opleveren, maar dan moeten die agenten wel op een kosten-efficiënte manier ontwikkeld kunnen worden. Eén manier om kosten te drukken is het hergebruik van eerder ontwikkelde (componenten van) agenten. Een belangrijke voorwaarde voor hergebruik van ontwikkelde componenten is een juiste specificatie van hun inhoud, zodat ze op basis daarvan later weer geselecteerd kunnen worden. In dit hoofdstuk doen we het voorstel om componenten van cognitieve agenten te specificeren aan de hand van de cognitieve capaciteiten die ze bezitten. Om dit gestandaardiseerd te doen, hebben we een structuur ontwikkeld om cognitieve capaciteiten te beschrijven en wel aan de hand van hun eigenschappen. We demonstreren de structuur door een aantal van de componenten van de beeldopbouwagent uit hoofdstuk 3 te beschrijven.

In **hoofdstuk 7** bespreken we hoe cognitieve modellen gebruikt kunnen worden om de rol van instructeur in een simulatietraining door een intelligente agent te laten vervullen. De taak van een instructeur is het geven van terugkoppeling op het taakgedrag van een student, zodat deze zijn taakkennis kan vergroten. Een voorwaarde voor het geven van zo'n terugkoppeling is het maken van een inschatting van de correctheid waarmee de student zijn taak uitvoert. Zo'n inschatting maakt een instructeur door het gedrag van de student te observeren en deze te relateren aan zijn kennis over juist en foutief gedrag.

Om dit gedrag van de instructeur te modelleren, maken we gebruik van cognitieve modellen. In het bijzonder hebben we in dit hoofdstuk een intelligente agent ontwikkeld die gegevens verzamelt over het taakgedrag van de student en deze vergelijkt met het gedrag van een verzameling cognitieve modellen. Eén daarvan is een expertmodel die de taak correct uitvoert; de andere cognitieve modellen zijn in meer of mindere mate onjuist (bijv. doordat er bepaalde essentiële taakkennis mist, of doordat er onjuiste redeneringen in gebruikt worden) en leiden mogelijk tot foutief gedrag. Aan de hand van deze vergelijkingen komt de agent tot een inschatting van het taakgedrag van de student en geeft op basis daarvan een bijpassende terugkoppeling. Om onze intelligente agent te toetsen hebben we gebruik gemaakt van gesimuleerde studenten, die elk een bepaald type fout representeerden. Uit de evaluatie bleek dat de agent praktisch alle gemaakte denkfouten juist kon diagnostiseren.

In **hoofdstuk 8** tenslotte vatten we het onderzoek uit dit proefschrift samen en trekken we conclusies over de toepassingsmogelijkheden van cognitieve modellen voor trainingssimulaties. Tevens inventariseren we welke vervolgonderzoeken nodig zijn, zowel voor de korte als voor de lange termijn.

Dankwoord

Nu mijn proefschrift af is zijn er mensen die me geïnteresseerd de notoire vraag stellen: “Waar zie je jezelf over vijf jaar?” Als deze vraag me vijf jaar geleden gesteld was, dan had ik waarschijnlijk geantwoord met een beeld dat overeenkomt met mijn huidige situatie: gepromoveerd en aan het werk bij een leuk wetenschappelijk onderzoeksinstituut. Wat ik vijf jaar geleden niet had kunnen bedenken zijn de mensen, die ik in die periode zou ontmoeten en de inzichten, die ik op zou doen.

Allereerst wil ik de twee mensen bedanken die mij gedurende het hele traject hebben begeleid: mijn promotor Jan Treur van de Vrije Universiteit en mijn co-promotor Karel van den Bosch van TNO. Van Jan heb ik, met vallen en opstaan, geleerd dat je wetenschappelijk onderzoek niet alleen hoeft te doen. De Annerieke die binnenkwam dacht vooral aan te moeten tonen dat ze zelf in staat was tot allerlei grootse zaken; mijn huidige ik ziet veel meer het belang van goede samenwerking en hoe je samen verder komt dan alleen. Karel is degene die mij vanaf het begin heeft ondersteund op een manier die ik als van een vader zou willen omschrijven. Altijd betrokken, met oog voor niet alleen de wetenschappelijke kant van mijn onderzoek, maar vooral ook voor mij. Daarnaast heb ik veel geleerd van Karels kritische blik ten opzichte van geschreven tekst. Als u dit proefschrift goed leesbaar vindt, dan weet u wie hiervoor te danken.

Als derde wil ik Catholijn Jonker bedanken en niet alleen voor haar rol als lid van de leescommissie. Toen ik op een gegeven moment tijdens mijn promotietraject niet zag hoe ik de eindstreep zou kunnen bereiken was het haar chargerende opmerking in de trant van ‘misschien kan je er beter mee stoppen’ die me liet voelen dat dat geen optie was. Het is mede aan haar coaching te danken dat dit proefschrift nu voor u ligt. Ook Michel Klein, mijn tweede co-promotor, heeft daaraan bijgedragen. Ik waardeer Michel erg voor de betrokkenheid waarmee hij zich opgesteld heeft, ondanks dat hij pas laat bij mijn promotie betrokken is geraakt. In dit rijtje wil ik ook Egon van den Broek noemen die mij juist in de beginperiode begeleid heeft en vanuit zijn doelgerichtheid een inspirerend voorbeeld was.

Naast Jan en Michel wil ik alle co-auteurs bedanken van de papers die in dit proefschrift zijn opgenomen: Fiemke Griffioen-Both, Rianne van Lambalgen, Tijmen Muller, Tina Mioch en Willem van Doesburg. Fiemke en Tina, ik had me geen betere masterstudenten kunnen wensen.

I am very grateful to the members of the reading committee: Catholijn Jonker, Cristiano Castelfranchi, Frank van Harmelen, Johan Hoorn, John-Jules Meyer, Mark Neerincx, and Wayne Gray. Thank you for spending some of your precious time on reading my dissertation, and on providing me with valuable comments.

Dit proefschrift was er niet geweest zonder de samenwerking tussen de VU en TNO die mij in staat heeft gesteld wetenschappelijk onderzoek te doen voor een concrete toepassing. Ik ben altijd dankbaar geweest voor deze promotieconstructie waarbij ik naast vier dagen op TNO, gemiddeld ook een dag in de week op de VU werkzaam was.

Van de VU wil ik alle collega's van de AI afdeling en vooral van de Agent Systems Research groep bedanken voor de gezelligheid tijdens mijn dagen daar, alsook tijdens conferenties in het buitenland. Vanuit de beginperiode wil ik met name Alexei Sharpanskykh, Mark van Assem, Mark Hoogendoorn, en Tibor Bosse bedanken; voor het einde 'de dames'. Via de VU heb ik ook deelgenomen aan SIKS, the School for Information Knowledge Systems. Alle sprekers die daar cursussen hebben gegeven en medepromovendi met wie ik eindeloos nageborreld en geweerwolfd heb: bedankt.

Van TNO wil ik vooral alle (voormalige) collega's van de afdeling Training en Instructie bedanken, waaronder in het bijzonder Anne Helsdingen, Jan Lubbers, Maaike Harbers, Nicolet Theunissen, Rob Verkuylen, Tijmen, Tina en op nummer 1 mijn paranimf Willem met wie ik vijf jaar lang zoveel meer dan een kamer heb gedeeld. Ook wil ik alle medepromovendi van TNO bedanken voor de gezellige etentjes en waardevolle gesprekken. Van die groep wil ik Peter-Paul van Maanen extra bedanken: ik had nooit zoveel van Amerika gezien als ik naast het kaartlezen ook de auto had moeten besturen.

Tijdens mijn promotie heb ik regelmatig geroepen dat ik "hartstikke ambitieus ben, alleen ook in veel dingen naast mijn onderzoek!" Mijn vriendschappen zijn daar wel het concreetste voorbeeld van. Allereerst wil ik 'de Needse Meiden' Annemieke, Daniëlle, Frytzen en Hester bedanken. Ik ken jullie al heel lang en hoop dat nog veel langer te blijven doen. Ten tweede mijn vrienden vanuit mijn oude CKI studie: Carola, Eva, Heleen, Jannie, Maarten, Marieke, Martijn, Pieter, Pim, Rommert en Stefan. Ik ben ontzettend blij met jullie. Paranimf Michiel, jij hoort natuurlijk ook in dat rijtje thuis, dank voor je onvoorwaardelijke vriendschap. Ten derde wil ik 'de Nieuwe Meiden' bedanken voor het delen van zoveel de laatste anderhalf jaar. Eefje, Eva, Maaike, Marieke, Marike en Saar, hoe bijzonder om jullie in zo'n korte tijd zo goed te leren kennen. Buiten alle kaders vallend wil ik Eveline, Jeanine en Lisette bedanken: jullie betekenen erg veel voor me.

Als laatste dank aan iedereen die hier niet expliciet genoemd staat, maar met wie ik de afgelopen jaren heb gelachen en een fijne tijd heb gehad.

Vijf jaar geleden had ik weinig zicht op hoe alles zou gaan lopen en wat mij zou brengen tot waar ik nu ben. Eén ding had ik op dat moment gelukkig wel al kunnen benoemen: de onvoorwaardelijke steun van mijn familie. MaPa, dank voor alle luisterende oren, peptalks, lieve kaartjes, bloemen-uit-de-bloementuin-meebrengende bezoekjes en heerlijke maaltijden-met-groenten-uit-de-groentetuin-gevulde Needse weekendjes. Het is fantastisch om me gekend en zo geliefd te weten. Henk Jan en Marike, prachtstel dat jullie zijn, ik hoop dat wij nog lang van elkaar mogen genieten. Jantine, dearest sister, wat ben ik toch blij met jou. Dr. worden is mooi, maar zussen zijn het allermooist.

SIKS Dissertatiereeks

1998

1998-1 Johan van den Akker (CWI)	DEGAS - An Active, Temporal Database of Autonomous Objects
1998-2 Floris Wiesman (UM)	Information Retrieval by Graphically Browsing Meta-Information
1998-3 Ans Steuten (TUD)	A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective
1998-4 Dennis Breuker (UM)	Memory versus Search in Games
1998-5 E.W.Oskamp (RUL)	Computerondersteuning bij Straftoemeting

1999

1999-1 Mark Sloof (VU)	Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products
1999-2 Rob Potharst (EUR)	Classification using decision trees and neural nets
1999-3 Don Beal (UM)	The Nature of Minimax Search
1999-4 Jacques Penders (UM)	The practical Art of Moving Physical Objects
1999-5 Aldo de Moor (KUB)	Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems
1999-6 Niek J.E. Wijngaards (VU)	Re-design of compositional systems
1999-7 David Spelt (UT)	Verification support for object database design
1999-8 Jacques H.J. Lenting (UM)	Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.

2000

2000-1 Frank Niessink (VU)	Perspectives on Improving Software Maintenance
2000-2 Koen Holtman (TUE)	Prototyping of CMS Storage Management
2000-3 Carolien M.T. Metselaar (UVA)	Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.
2000-4 Geert de Haan (VU)	ETAG, A Formal Model of Competence Knowledge for User Interface Design
2000-5 Ruud van der Pol (UM)	Knowledge-based Query Formulation in Information Retrieval.
2000-6 Rogier van Eijk (UU)	Programming Languages for Agent Communication
2000-7 Niels Peek (UU)	Decision-theoretic Planning of Clinical Patient Management
2000-8 Veerle Coup (EUR)	Sensitivity Analyis of Decision-Theoretic Networks
2000-9 Florian Waas (CWI)	Principles of Probabilistic Query Optimization
2000-10 Niels Nes (CWI)	Image Database Management System Design Considerations, Algorithms and Architecture
2000-11 Jonas Karlsson (CWI)	Scalable Distributed Data Structures for Database Management

2001

2001-1 Silja Renooij (UU)	Qualitative Approaches to Quantifying Probabilistic Networks
2001-2 Koen Hindriks (UU)	Agent Programming Languages: Programming with Mental Models
2001-3 Maarten van Someren (UvA)	Learning as problem solving
2001-4 Evgueni Smirnov (UM)	Conjunctive and Disjunctive Version Spaces with

	Instance-Based Boundary Sets
2001-5 Jacco van Ossenbruggen (VU)	Processing Structured Hypermedia: A Matter of Style
2001-6 Martijn van Welie (VU)	Task-based User Interface Design
2001-7 Bastiaan Schonhage (VU)	Diva: Architectural Perspectives on Information Visualization
2001-8 Pascal van Eck (VU)	A Compositional Semantic Structure for Multi-Agent Systems Dynamics.
2001-9 Pieter Jan 't Hoen (RUL)	Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes
2001-10 Maarten Sierhuis (UvA)	Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design
2001-11 Tom M. van Engers (VUA)	Knowledge Management: The Role of Mental Models in Business Systems Design

2002

2002-01 Nico Lassing (VU)	Architecture-Level Modifiability Analysis
2002-02 Roelof van Zwol (UT)	Modelling and searching web-based document collections
2002-03 Henk Ernst Blok (UT)	Database Optimization Aspects for Information Retrieval
2002-04 Juan Roberto Castelo Valdueza (UU)	The Discrete Acyclic Digraph Markov Model in Data Mining
2002-05 Radu Serban (VU)	The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents
2002-06 Laurens Mommers (UL)	Applied legal epistemology; Building a knowledge-based ontology of the legal domain
2002-07 Peter Boncz (CWI)	Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications
2002-08 Jaap Gordijn (VU)	Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas
2002-09 Willem-Jan van den Heuvel(KUB)	Integrating Modern Business Applications with Objectified Legacy Systems
2002-10 Brian Sheppard (UM)	Towards Perfect Play of Scrabble
2002-11 Wouter C.A. Wijngaards (VU)	Agent Based Modelling of Dynamics: Biological and Organisational Applications
2002-12 Albrecht Schmidt (UVA)	Processing XML in Database Systems
2002-13 Hongjing Wu (TUE)	A Reference Architecture for Adaptive Hypermedia Applications
2002-14 Wieke de Vries (UU)	Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems
2002-15 Rik Eshuis (UT)	Semantics and Verification of UML Activity Diagrams for Workflow Modelling
2002-16 Pieter van Langen (VU)	The Anatomy of Design: Foundations, Models and Applications
2002-17 Stefan Manegold (UVA)	Understanding, Modeling, and Improving Main-Memory Database Performance

2003

2003-01 Heiner Stuckenschmidt (VU)	Ontology-Based Information Sharing in Weakly Structured Environments
2003-02 Jan Broersen (VU)	Modal Action Logics for Reasoning About Reactive Systems
2003-03 Martijn Schuemie (TUD)	Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy
2003-04 Milan Petkovic (UT)	Content-Based Video Retrieval Supported by Database Technology
2003-05 Jos Lehmann (UVA)	Causation in Artificial Intelligence and Law - A modelling approach
2003-06 Boris van Schooten (UT)	Development and specification of virtual environments
2003-07 Machiel Jansen (UvA)	Formal Explorations of Knowledge Intensive Tasks
2003-08 Yongping Ran (UM)	Repair Based Scheduling
2003-09 Rens Kortmann (UM)	The resolution of visually guided behaviour
2003-10 Andreas Lincke (UvT)	Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture
2003-11 Simon Keizer (UT)	Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks
2003-12 Roeland Ordelman (UT)	Dutch speech recognition in multimedia information retrieval
2003-13 Jeroen Donkers (UM)	Nosce Hostem - Searching with Opponent Models

2003-14 Stijn Hoppenbrouwers (KUN) Freezing Language: Conceptualisation Processes across ICT-Supported Organisations
2003-15 Mathijs de Weerdt (TUD) Plan Merging in Multi-Agent Systems
2003-16 Menzo Windhouwer (CWI) Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses
2003-17 David Jansen (UT) Extensions of Statecharts with Probability, Time, and Stochastic Timing
2003-18 Levente Kocsis (UM) Learning Search Decisions

2004

2004-01 Virginia Dignum (UU) A Model for Organizational Interaction: Based on Agents, Founded in Logic
2004-02 Lai Xu (UvT) Monitoring Multi-party Contracts for E-business
2004-03 Perry Groot (VU) A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving
2004-04 Chris van Aart (UVA) Organizational Principles for Multi-Agent Architectures
2004-05 Viara Popova (EUR) Knowledge discovery and monotonicity
2004-06 Bart-Jan Hommes (TUD) The Evaluation of Business Process Modeling Techniques
2004-07 Elise Boltjes (UM) Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes
2004-08 Joop Verbeek(UM) Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise
2004-09 Martin Caminada (VU) For the Sake of the Argument; explorations into argument-based reasoning
2004-10 Suzanne Kabel (UVA) Knowledge-rich indexing of learning-objects
2004-11 Michel Klein (VU) Change Management for Distributed Ontologies
2004-12 The Duy Bui (UT) Creating emotions and facial expressions for embodied agents
2004-13 Wojciech Jamroga (UT) Using Multiple Models of Reality: On Agents who Know how to Play
2004-14 Paul Harrenstein (UU) Logic in Conflict. Logical Explorations in Strategic Equilibrium
2004-15 Arno Knobbe (UU) Multi-Relational Data Mining
2004-16 Federico Divina (VU) Hybrid Genetic Relational Search for Inductive Learning
2004-17 Mark Winands (UM) Informed Search in Complex Games
2004-18 Vania Bessa Machado (UvA) Supporting the Construction of Qualitative Knowledge Models
2004-19 Thijs Westerveld (UT) Using generative probabilistic models for multimedia retrieval
2004-20 Madelon Evers (Nyenrode) Learning from Design: facilitating multidisciplinary design teams

2005

2005-01 Floor Verdenius (UVA) Methodological Aspects of Designing Induction-Based Applications
2005-02 Erik van der Werf (UM)) AI techniques for the game of Go
2005-03 Franc Grootjen (RUN) A Pragmatic Approach to the Conceptualisation of Language
2005-04 Nirvana Meratnia (UT) Towards Database Support for Moving Object data
2005-05 Gabriel Infante-Lopez (UVA) Two-Level Probabilistic Grammars for Natural Language Parsing
2005-06 Pieter Spronck (UM) Adaptive Game AI
2005-07 Flavius Frasincar (TUE) Hypermedia Presentation Generation for Semantic Web Information Systems
2005-08 Richard Vdovjak (TUE) A Model-driven Approach for Building Distributed Ontology-based Web Applications
2005-09 Jeen Broekstra (VU) Storage, Querying and Inferencing for Semantic Web Languages
2005-10 Anders Bouwer (UVA) Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments
2005-11 Elth Ogston (VU) Agent Based Matchmaking and Clustering - A Decentralized Approach to Search
2005-12 Csaba Boer (EUR) Distributed Simulation in Industry
2005-13 Fred Hamburg (UL) Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen
2005-14 Borys Omelayenko (VU) Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics
2005-15 Tibor Bosse (VU) Analysis of the Dynamics of Cognitive Processes
2005-16 Joris Graaumans (UU) Usability of XML Query Languages
2005-17 Boris Shishkov (TUD) Software Specification Based on Re-usable Business Components
2005-18 Danielle Sent (UU) Test-selection strategies for probabilistic networks

2005-19 Michel van Dartel (UM) Situated Representation
2005-20 Cristina Coteanu (UL) Cyber Consumer Law, State of the Art and Perspectives
2005-21 Wijnand Derks (UT) Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006

2006-01 Samuil Angelov (TUE) Foundations of B2B Electronic Contracting
2006-02 Cristina Chisalita (VU) Contextual issues in the design and use of information technology in organizations
2006-03 Noor Christoph (UVA) The role of metacognitive skills in learning to solve problems
2006-04 Marta Sabou (VU) Building Web Service Ontologies
2006-05 Cees Pierik (UU) Validation Techniques for Object-Oriented Proof Outlines
2006-06 Ziv Baida (VU) Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling
2006-07 Marko Smiljanic (UT) XML schema matching – balancing efficiency and effectiveness by means of clustering
2006-08 Eelco Herder (UT) Forward, Back and Home Again - Analyzing User Behavior on the Web
2006-09 Mohamed Wahdan (UM) Automatic Formulation of the Auditor's Opinion
2006-10 Ronny Siebes (VU) Semantic Routing in Peer-to-Peer Systems
2006-11 Joeri van Ruth (UT) Flattening Queries over Nested Data Types
2006-12 Bert Bongers (VU) Interactivation - Towards an e-cology of people, our technological environment, and the arts
2006-13 Henk-Jan Lebbink (UU) Dialogue and Decision Games for Information Exchanging Agents
2006-14 Johan Hoorn (VU) Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
2006-15 Rainer Malik (UU) CONAN: Text Mining in the Biomedical Domain
2006-16 Carsten Riggelsen (UU) Approximation Methods for Efficient Learning of Bayesian Networks
2006-17 Stacey Nagata (UU) User Assistance for Multitasking with Interruptions on a Mobile Device
2006-18 Valentin Zhizhkun (UVA) Graph transformation for Natural Language Processing
2006-19 Birna van Riemsdijk (UU) Cognitive Agent Programming: A Semantic Approach
2006-20 Marina Velikova (UvT) Monotone models for prediction in data mining
2006-21 Bas van Gils (RUN) Aptness on the Web
2006-22 Paul de Vrieze (RUN) Fundaments of Adaptive Personalisation
2006-23 Ion Juvina (UU) Development of Cognitive Model for Navigating on the Web
2006-24 Laura Hollink (VU) Semantic Annotation for Retrieval of Visual Resources
2006-25 Madalina Drugan (UU) Conditional log-likelihood MDL and Evolutionary MCMC
2006-26 Vojkan Mihajlovic (UT) Score Region Algebra: A Flexible Framework for Structured Information Retrieval
2006-27 Stefano Bocconi (CWI) Vox Populi: generating video documentaries from semantically annotated media repositories
2006-28 Borkur Sigurbjornsson (UVA) Focused Information Access using XML Element Retrieval

2007

2007-01 Kees Leune (UvT) Access Control and Service-Oriented Architectures
2007-02 Wouter Teepe (RUG) Reconciling Information Exchange and Confidentiality: A Formal Approach
2007-03 Peter Mika (VU) Social Networks and the Semantic Web
2007-04 Jurriaan van Diggelen (UU) Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach
2007-05 Bart Schermer (UL) Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance
2007-06 Gilad Mishne (UVA) Applied Text Analytics for Blogs
2007-07 Natasa Jovanovic' (UT) To Whom It May Concern - Addressee Identification in Face-to-Face Meetings
2007-08 Mark Hoogendoorn (VU) Modeling of Change in Multi-Agent Organizations
2007-09 David Mobach (VU) Agent-Based Mediated Service Negotiation
2007-10 Huib Aldewereld (UU) Autonomy vs. Conformity:

an Institutional Perspective on Norms and Protocols
2007-11 Natalia Stash (TUE) Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System
2007-12 Marcel van Gerven (RUN) Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty
2007-13 Rutger Rienks (UT) Meetings in Smart Environments; Implications of Progressing Technology
2007-14 Niek Bergboer (UM) Context-Based Image Analysis
2007-15 Joyca Lacroix (UM) NIM: a Situated Computational Memory Model
2007-16 Davide Grossi (UU) Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems
2007-17 Theodore Charitos (UU) Reasoning with Dynamic Networks in Practice
2007-18 Bart Orriens (UvT) On the development an management of adaptive business collaborations
2007-19 David Levy (UM) Intimate relationships with artificial partners
2007-20 Slinger Jansen (UU) Customer Configuration Updating in a Software Supply Network
2007-21 Karianne Vermaas (UU) Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005
2007-22 Zlatko Zlatev (UT) Goal-oriented design of value and process models from patterns
2007-23 Peter Barna (TUE) Specification of Application Logic in Web Information Systems
2007-24 Georgina Ramrez Camps (CWI) Structural Features in XML Retrieval
2007-25 Joost Schalken (VU) Empirical Investigations in Software Process Improvement

2008

2008-01 Katalin Boer-Sorbn (EUR) Agent-Based Simulation of Financial Markets: A modular,continuous-time approach
2008-02 Alexei Sharpanskykh (VU) On Computer-Aided Methods for Modeling and Analysis of Organizations
2008-03 Vera Hollink (UVA) Optimizing hierarchical menus: a usage-based approach
2008-04 Ander de Keijzer (UT) Management of Uncertain Data - towards unattended integration
2008-05 Bela Mutschler (UT) Modeling and simulating causal dependencies on process-aware information systems from a cost perspective
2008-06 Arjen Hommersom (RUN) On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective
2008-07 Peter van Rosmalen (OU) Supporting the tutor in the design and support of adaptive e-learning
2008-08 Janneke Bolt (UU) Bayesian Networks: Aspects of Approximate Inference
2008-09 Christof van Nimwegen (UU) The paradox of the guided user: assistance can be counter-effective
2008-10 Wauter Bosma (UT) Discourse oriented summarization
2008-11 Vera Kartseva (VU) Designing Controls for Network Organizations: A Value-Based Approach
2008-12 Jozsef Farkas (RUN) A Semiotically Oriented Cognitive Model of Knowledge Representation
2008-13 Caterina Carraciolo (UVA) Topic Driven Access to Scientific Handbooks
2008-14 Arthur van Bunningen (UT) Context-Aware Querying; Better Answers with Less Effort
2008-15 Martijn van Otterlo (UT) The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains.
2008-16 Henriette van Vugt (VU) Embodied agents from a user's perspective
2008-17 Martin Op 't Land (TUD) Applying Architecture and Ontology to the Splitting and Allying of Enterprises
2008-18 Guido de Croon (UM) Adaptive Active Vision
2008-19 Henning Rode (UT) From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search
2008-20 Rex Arendsen (UVA) Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven
2008-21 Krisztian Balog (UVA) People Search in the Enterprise
2008-22 Henk Koning (UU) Communication of IT-Architecture
2008-23 Stefan Visscher (UU) Bayesian network models for the management of ventilator-associated pneumonia
2008-24 Zharko Aleksovski (VU) Using background knowledge in ontology matching
2008-25 Geert Jonker (UU) Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency

2008-26 Marijn Huijbregts (UT) Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled
2008-27 Hubert Vogten (OU) Design and Implementation Strategies for IMS Learning Design
2008-28 Ildiko Flesch (RUN) On the Use of Independence Relations in Bayesian Networks
2008-29 Dennis Reidsma (UT) Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans
2008-30 Wouter van Atteveldt (VU) Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content
2008-31 Loes Braun (UM) Pro-Active Medical Information Retrieval
2008-32 Trung H. Bui (UT) Toward Affective Dialogue Management using Partially Observable Markov Decision Processes
2008-33 Frank Terpstra (UVA) Scientific Workflow Design; theoretical and practical issues
2008-34 Jeroen de Knijf (UU) Studies in Frequent Tree Mining
2008-35 Ben Torben Nielsen (UvT) Dendritic morphologies: function shapes structure

2009

2009-01 Rasa Jurgelenaite (RUN) Symmetric Causal Independence Models
2009-02 Willem Robert van Hage (VU) Evaluating Ontology-Alignment Techniques
2009-03 Hans Stol (UvT) A Framework for Evidence-based Policy Making Using IT
2009-04 Josephine Nabukenya (RUN) Improving the Quality of Organisational Policy Making using Collaboration Engineering
2009-05 Sietse Overbeek (RUN) Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality
2009-06 Muhammad Subianto (UU) Understanding Classification
2009-07 Ronald Poppe (UT) Discriminative Vision-Based Recovery and Recognition of Human Motion
2009-08 Volker Nannen (VU) Evolutionary Agent-Based Policy Analysis in Dynamic Environments
2009-09 Benjamin Kanagwa (RUN) Design, Discovery and Construction of Service-oriented Systems
2009-10 Jan Wielemaker (UVA) Logic programming for knowledge-intensive interactive applications
2009-11 Alexander Boer (UVA) Legal Theory, Sources of Law & the Semantic Web
2009-12 Peter Massuthe (TUE) Operating Guidelines for Services
2009-13 Steven de Jong (UM) Fairness in Multi-Agent Systems
2009-14 Maksym Korotkiy (VU) From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)
2009-15 Rinke Hoekstra (UVA) Ontology Representation - Design Patterns and Ontologies that Make Sense
2009-16 Fritz Reul (UvT) New Architectures in Computer Chess
2009-17 Laurens van der Maaten (UvT) Feature Extraction from Visual Data
2009-18 Fabian Groffen (CWI) Armada, An Evolving Database System
2009-19 Valentin Robu (CWI) Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets
2009-20 Bob van der Vecht (UU) Adjustable Autonomy: Controling Influences on Decision Making
2009-21 Stijn Vanderlooy (UM) Ranking and Reliable Classification
2009-22 Pavel Serdyukov (UT) Search For Expertise: Going beyond direct evidence
2009-23 Peter Hofgesang (VU) Modelling Web Usage in a Changing Environment
2009-24 Annerieke Heuvelink (VU) **Cognitive Models for Training Simulations**