Cognitive Models of Strategy Shifts in Interactive Behavior

Christian Pieter Janssen
cjanssen@ai.rug.nl
July 7th 2008

Internal supervisor:
Dr. Hedderik van Rijn
University of Groningen
Department of Artificial Intelligence

External supervisor:
Prof. Dr. Wayne D. Gray
Rensselaer Polytechnic Institute
Department of Cognitive Science

Master Human-Machine Communication
Department of Artificial Intelligence
University of Groningen

Abstract

In this study, we use cognitive models to investigate how humans develop preferences for specific strategies in interactive behavior. In our case study, subjects show a shift in strategy, to strategies that are mostly based either on interaction with the environment or on memory. ACT-R 5 models are unable to show this effect. This is ascribed to its utility learning mechanism. We use ACT-R 6, which has a new utility learning mechanism. Results indicate that regular ACT-R 6 models (i.e., models without changes in the architecture) can provide a better fit to the human data than regular ACT-R 5 models. However, the ACT-R 6 models show equal or less good fits when compared to ACT-R 5 models with changes in the architecture. In general, the goodness of fit of the models depends on the modeler's conception of rewards: what is the magnitude of the reward (and to what concept of the task is this magnitude related), and when does the model receive rewards? These concerns are new for ACT-R researchers, as in ACT-R 5 a modeler was limited in the type of rewards they could give to their model. General implications of this observation are discussed.

Acknowledgements

This thesis is the final product of my years as a student at the University of Groningen. It is a special moment, as it is the end of an era for me. It thus feels appropriate to thank some of the people who helped me to reach this point.

First of all I want to thank the members of the CogWorks laboratory of the Rensselaer Polytechnic Institute, with whom I worked the last few months. Their warm welcome and support made me feel at home from the day I entered the lab until the day that I left. I especially want to thank my advisor Wayne Gray for his guidance in this project, for the good conversations we had and for taking the time to supervise yet another student. I also want to thank Michael Schoelles. Without his help it would have been a lot harder to understand the programming language Lisp.

I also want to thank all the members of the Artificial Intelligence department and the Cognitive Modeling group at the University of Groningen. Their enthusiasm during courses, projects and collaborative initiatives awakened my enthusiasm and passion for cognitive science. I especially want to thank Hedderik van Rijn, who has been my advisor for several years, and who has always been a source of support and feedback.

The “Stichting Groninger UniversiteitsFonds” and the “Marco Polo Fund” support me financially. A big thanks to them, as this international experience would not have been possible without their help.

Also, a big thanks to the (former and current) students of student union CoVer (at the Artificial Intelligence department in Groningen). Together we made sure that our college years were some of the best years in our lives. Besides sharing fun, I also learned a lot from them during the many hours, weeks, months and years that we spend together organizing and taking on big events. I developed many skills by following their lead and by reflecting my experiences with them.

I want to thank my family and close friends for their support, and for keeping in touch despite the spatial and temporal difference between The Netherlands and the USA. A special thanks goes to Tobias who helped and encouraged me to chase my dreams.

Last of all, thank you, reader of this thesis, for being interested in my work. I hope you like it!

July 3rd, Troy New York

Table of Contents

Chapter 1 Introduction

1.1 How do we develop preferences for specific strategies?

Many tasks can be solved using different strategies. Based on previous experiences with those strategies, humans develop preferences for specific strategies in specific contexts. One example is the Blocks World task, studied by Gray, Sims, Fu and Schoelles (2006). We will formally introduce this task in Chapter 3, but will describe the main finding here. In this task, the ease at which task relevant information is available influences the strategies that humans use to perform the task. If information is easy to access, humans tend to interact a lot with the task environment to get the task information and to perform the task. However, if it is harder to access the information, they interact less with the environment. Instead, they tend to remember and use more task information at once.

Cognitive models of this task have been developed in ACT-R (more background on ACT-R will be provided in Chapter 5). However, these models fail to provide a good fit to human data (Gray et al., 2005, see also Chapter 6 of this thesis). This is ascribed to conditioning, the mechanism of the cognitive model that learns to prefer specific strategies. Recently, the mechanism for conditioning has changed (e.g., Anderson, 2007; Fu & Anderson, 2004, 2006), and we call the different versions of ACT-R: ACT-R 5 (old) and ACT-R 6 (new). The question arises if the models of the Blocks World task would benefit from this change.

In the meantime another modeling effort has been successful in providing a good fit to the human data (Gray et al., 2006). Although this model is no cognitive model, it does use techniques that are very similar to those that are now used in ACT-R 6. In this study we therefore investigate if cognitive models of the Blocks World task that are developed in ACT-R 6 provide a better fit to the human data than the previous developed ACT-R 5 models (Gray et al., 2005). If the models capture human behavior successfully, they can give more insight in general human behavior that involves strategy shifts.

1.2 Research question

The core question of this research is: "Does an ACT-R 6 model of the Blocks World task that incorporates the latest mechanism for conditioning (Anderson, 2007; Fu & Anderson, 2004, 2006) provide a better qualitative and quantitative fit to the human data than the ACT-R 5 models of that task (Gray et al., 2005)?"

To answer this question, the old models are reexamined, and implemented in ACT-R 6 (Anderson, 2007). ACT-R 6 models try to optimize their behavior as to maximize the rewards that they get from the environment. Both the moment at which a reward is given and the magnitude of the reward can have a big impact on the model's behavior. In ACT-R 5 the magnitude of rewards could not be varied (there were only successes or failures). In ACT-R 6 it can be varied, and it is an open question how the magnitude of the reward should be set. Our study is the first to investigate how the choice for the magnitude of the reward impacts a model's behavior.

1.3 Scientific relevance

The research will increase the insight in how strategy shifts occur in the human mind. By using cognitive models, interactive behavior (i.e., behavior in which humans interacts with their environment, for example with computers) that is observed at a macro level (i.e., a strategy shift), can be explained by processes at a millisecond level. That is to say, we are then able to explain how strategy selections are caused by components of the developed cognitive model. For the ACT-R community, the research provides a case study to prove the value of the new introduced reinforcement learning mechanism. It will also show how a modeler's choices for the reward magnitude and the moment at which a reward is given to a model impact the model's behavior.

1.4 Structure of the thesis

The thesis is structured as followed. Chapter 2 gives the necessary theoretical background on strategy selection in interactive behavior. We describe in what types of strategies we are interested, and how we can model those. In Chapter 3 we discuss the Blocks World task, our case study. After giving a general overview of the task, we report experimental data that will form the benchmark with which we can test our models.

We then discuss three efforts for modeling the Blocks World task. The first effort, reported in Chapter 4, describes what humans would do if they behaved as an ideal performer model (Gray et al., 2006): a model that optimizes its behavior given the constraints that are put on its performance due to the environment in which it acts and physical and cognitive limitations. As we will see, this model is able to provide a good fit to the human data. However, it does not necessarily perform the task in a "human way".

Our next two models use the cognitive architecture ACT-R. These chapters are preceded by a general introduction of ACT-R in Chapter 5. In Chapter 6 we will briefly describe the ACT-R 5 models of the Blocks World task (Gray et al., 2005). As we will see, these models do not provide a good fit to the human data. We will then discuss the main work of the current thesis: the models that are developed in ACT-R 6. Chapter 7 will describe the structure of these models and the results. The models provide better fits than regular ACT-R 5 models, and provide valuable insight in how a modeler's choice for the magnitude of the reward and the moment in time at which rewards are given impact the model's behavior.

In Chapter 8 we summarize our findings. We will present specific conclusions for the ACT-R community and for other studies of strategy shifts in interactive behavior. The thesis is followed by a set of Appendixes in which more details can be found about ACT-R, the structure of the models and the performance of our models. Also some recommendations for future work are included. By the end, the reader will know how strategy shifts can be modeled in ACT-R 5 and ACT-R 6. In addition, the reader will know how the modeler's choices for the magnitude of rewards and the moment in time at which rewards are given in ACT-R 6 models influence the model's behavior.

Chapter 2 Theoretical section: Strategy selection and strategy shifts in interactive behavior

2.1 Decomposing tasks in interactive routines

Many tasks can be decomposed in smaller sub tasks. Take the simple example task of making coffee, which already requires a lot of sub tasks such as getting a filter, putting the filter in the coffee machine and putting coffee in the filter (Larkin, 1989). If these tasks involve interaction with the environment (e.g., interacting with our coffee machine), and if the environment (or the device) can influence behavior, we will call this interactive behavior. When performing interactive behavior, we use basic perceptual, motor and cognitive operators (Card, Moran, & Newell, 1983). For example, we *look for* the coffee container, *open* it, and *think* about how much coffee we want to put into the filter. These basic operators can be combined in different manners to form microstrategies or interactive routines, which take about one-third to three seconds to complete (Gray & Boehm-Davis, 2000; Gray et al., 2006).

Different combinations of basic operators form different interactive routines. Each basic operator needs a different time to perform, and as a result each interactive routine will take a different time to complete. It has been shown that accumulation of these differences, sometimes as small as a few milliseconds, influences overall task performance drastically (Gray & Boehm-Davis, 2000; Gray et al., 2006). Which interactive routines are used for performing a task can be influenced by the interface of a device. One of the first tasks in which this has been demonstrated is the Blocks World task (Ballard, Hayhoe, & Pelz, 1995; Ballard et al., 1997). This study showed that if the availability of task information is shifted from requiring only an eye-movement to requiring a head-movement, subjects tend to remember and use more task information at a time. Thus, a change in the interface enforces the use of different interactive routines and these different interactive routines result in the use of different strategies: different amounts of information were remembered and used at a time.

Strategy selection is the process of choosing to apply a specific strategy (or a combination of interactive routines). A strategy shift is the process of learning to favor the use of a specific strategy over others, by exploring alternatives. In this thesis, we will try to explain strategy shifts in terms of the usefulness and experience with different interactive routines.

2.2 Minimum memory or soft constraints?

The ease of access to information can influence the selection of strategies, at least in the Blocks World task (Ballard et al., 1995; Ballard et al., 1997; Gray et al., 2006). But how can this choice in strategy be explained? At least two different theories exist, which we will now discuss. The first theory states that people try to limit the amount of information that they keep in memory by interacting with their environment. According to this theory, instead of keeping very detailed mental representations of the environment in our mind, we directly manipulate the world itself, as the world is “its own best model” (Brooks, 1991). The foundation of this theory can be found in the field of embodied cognition (e.g., Wilson, 2002). Motivation for this theory is that the amount of resources that our memory has available to store or process information is limited (e.g., Miller, 1956). People therefore try to avoid overloading it and to keep it free for more important tasks. We will refer to this theory as the minimum memory hypothesis (in line with Gray et al., 2006), because it states that we try to keep the amount of memory used for performance to a minimum.

The second theory is the soft constraints hypothesis (Gray et al., 2006). According to this theory, people use interactive routines to perform interactive behavior. Sometimes, people can choose between multiple interactive routines to perform at a given stage of the task. The soft constraints hypothesis predicts that people (who have some prior experience) will then choose the interactive routine that takes the least amount of time, but that still leads to a reasonable performance. Thus, interactive routines are chosen so as to optimize the amount of time spend on each of the interactive routines. However, this is not a hard constraint that governs behavior all the time. Rather, other factors can override the choice for the right interactive routines. Examples are training or deliberately choosing different strategies (Gray et al., 2006).

Given the costs of different interactive routines, different strategies can be used. On the one extreme, people might limit their use of memory and use interaction with their environment to reach their goal. We will refer to these types of strategies as interaction-intensive strategies. On the other extreme are strategies that limit the interaction with the environment, and rather use memory to store a lot of information. We will refer to these strategies as memory-intensive strategies. In between these two extremes can be different strategies that are more or less memory or interaction-intensive.

2.3 Alternative accounts of strategy selection

We framed strategy selection in the context of interactive behavior. Our definition is in line with alternative definitions that are not necessarily concerned with interactive behavior. Strategy selection always involves a choice amongst alternatives for working towards completing a goal (e.g., Erev & Barron, 2005; Gonzalez, Lerch, & Lebiere, 2003; Lovett, 1998, 2005; Rieskamp & Otto, 2006; Roberts & Newton, 2001; Siegler, 1991; Sperling & Dosher, 1986). What the alternatives (or individual strategies) are differs for each specific goal. Different theories also have different ways in which the strategies are represented (see Sperling & Dosher, 1986, for a discussion of some alternatives). In the cognitive architecture ACT-R (Anderson, 2007; Anderson et al., 2004), in which our theories are grounded, strategies are most of the time modeled in the form of procedural knowledge (Lovett, 1998). We will discuss this in more detail in Chapter 5. Alternative attempts have been made to model strategy selection based on the retrieval of declarative knowledge (Gonzalez et al., 2003). We will not further discuss this alternative theory.

All theories of strategy selection state that strategy selection is flexible. Different people can learn to favor different strategies given their own specific circumstances, and they can change their preferences. Different theories pose different ideas about how people learn to favor specific strategies. However, all theories require feedback on the selection of strategies in order to optimize the choice (e.g., Erev & Barron, 2005; Gonzalez et al., 2003; Lovett, 1998, 2005; Rieskamp & Otto, 2006; Roberts & Newton, 2001; Siegler, 1991; Sperling & Dosher, 1986). In the upcoming chapters we will discuss different mechanisms for learning to favor specific strategies.

2.4 Modeling strategy selection

Human strategy selection can be modeled using cognitive architectures. Cognitive architectures give a functional explanation of how cognitive processes are achieved in the brain (Anderson, 2007). They provide the overall framework of human cognition in which specific theories of specific cognitive processes and tasks can be tested, by developing specific models of the task at hand. These theories and models need to be specified in terms of the architecture, which forces researches to make detailed specifications of their theory.

In Chapter 5 we will give a more detailed description of the cognitive architecture ACT-R (Anderson, 2007; Anderson et al., 2004) that is used for the current research, to be followed by a description of the developed models in Chapter 6 (based on work by Gray et al., 2005) and Chapter 7. Although the main work is in comparing the results of these two models, they are also compared to the results of an ideal performer model (Gray et al., 2006). Ideal performer models do not try to capture exactly *how* humans perform a task. Rather, they try to capture what the best possible performance would be, given a performance evaluation criterion. This way the models place an upper boundary on the range of strategies that might be used by humans.

In an ideal performer model one has to identify the cognitive side conditions that limit human performance (Simon, 1992), such as memory capacity. An ideal performer model tries to capture the *optimal* or *ideal* performance given a specific task. The theory of rational analysis (Anderson, 1990, 1991) is used for this optimization process. A rational analysis first of all requires researchers to frame how information is processed, by identifying three aspects of the task: the goals that are pursued, the relevant structure of the environment in which those goals are pursued, and the costs of pursuing the goals (Anderson, 1991). Given this framing, optimal behavior is behavior in which one maximizes ones goals while at the same time minimizing the costs of achieving the goal (Anderson, 1991).

In an ideal performer model the side conditions place costs on performing a goal. For each interactive routine one can identify what its cost is given the side conditions that apply. If at one time there is a choice between different interactive routines, an ideal performer would select the interactive routine that minimizes the costs of the side conditions (Gray et al., 2006). In Chapter 4 of this thesis we will discuss a previous successful effort of developing an ideal performer model of performance in the Blocks World task (Gray et al., 2006). However, we will now first take a closer look at the Blocks World task, our case study for testing theories about human strategy selection.

Chapter 3: The Blocks World task

In this chapter we explain the structure of the Blocks World task. We start out by explaining the task, and its sub tasks. Then we identify the possible strategies. After this we discuss experimental results (Gray et al., 2006). The data of this study will provide the benchmark against which we compare the data that our models provide (see Chapters 4, 6, and 7).

3.1 Task structure and sub tasks of the Blocks World task

In the Blocks World task, humans are shown a pattern of eight colored blocks in a target window. They are told to replicate this pattern in a workspace window, by dragging blocks from a resource window (which contains blocks of each color) to the right position in the workspace window. Figure 3.1 shows these three windows (Gray et al., 2006). The target pattern is shown in a four by four grid and consists of eight blocks. Each block has one out of eight different colors, randomly chosen, and during each trial each color can be present in at most two blocks. In the normal experimental version of the Blocks World task all three windows are covered by gray rectangles. At each moment in time the information of at most one window can become visible by moving the mouse cursor into the window area. The resource window and workspace window become visible immediately after the mouse cursor enters the window area. In contrast, the information in the target window only becomes available after a lockout time has passed. This lockout time is manipulated between subjects and has values of 0, 200, 400, 800, 1600 and 3200 milliseconds (Gray et al., 2006).

The Blocks World task involves four different sub tasks, which are depicted in Figure 3.2. The first is called "start trial" and involves pressing the start button. This is followed by the sub task called "study blocks in target window". In this sub task a number of blocks that has not yet been placed in the target window is studied. After having studied a number of blocks, the blocks can then be placed in the target window. This is the sub task "place blocks in workspace window". In order to place blocks, the right color and location of studied blocks have to be retrieved from memory. Once this succeeds, a block of the right color needs to be clicked on in the resource window and then moved to the right position in the workspace window. Blocks can be placed for as long as someone has blocks encoded in memory and is able to retrieve them. If all encoded blocks are placed, or if a subject is unable to retrieve information about studied blocks from memory, the next sub task is chosen. If there are still blocks left to place, the next step is again the sub task "study blocks in target window". If all blocks have been placed, the sub task "stop trial" is executed: the stop button is pressed.

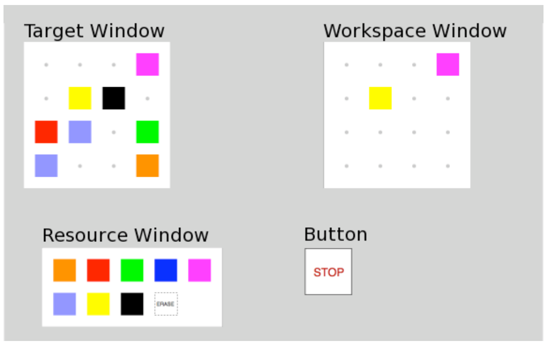

Figure 3.1: The task environment of the Blocks World task, with each of the four screen areas (windows) denoted in them.

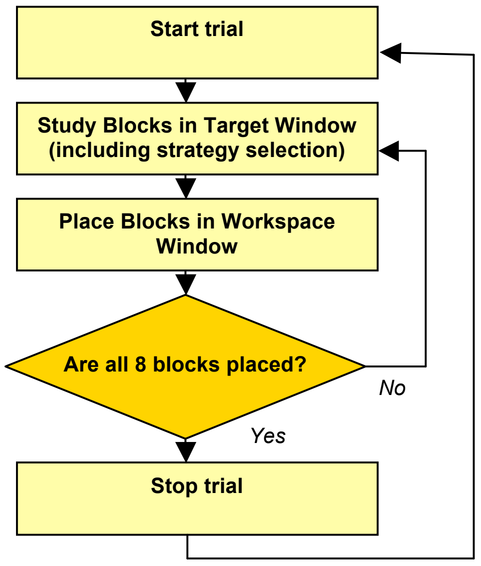
Start trial
Study Blocks in Target Window (including strategy selection)
Place Blocks in Workspace Window
Are all 8 blocks placed?
No
Yes
Stop trial

Figure 3.2: The four main sub tasks of the Blocks World task: start a trial, study blocks in the target window (including a strategy selection), place blocks in the workspace window and Stop Trial

3.2 Strategies in the Blocks World task

The basic strategy for solving the Blocks World task is selected in the sub task "study one or more blocks in the target window." We refer to a strategy as encode-x, where x denotes the number of blocks that will be studied during an upcoming visit to the target window. There are eight encode-x strategies, encode-1 until encode-8. The more blocks subjects place after each visit to the target window, the less visits they have to pay to the target window. As a result, the subjects have to perform two basic operators less often: moving visual attention and the mouse to the target window. They also have to wait less often for the lockout time. The fastest strategy would thus be a successful use of encode-8: to study all eight blocks at once, and then successfully place them after this study round.

In practice however, most humans will have problems in the majority of the trials with storing and retrieving all eight blocks from memory. Slower strategies would occur when not all blocks are placed after a study round, and the task is completed by multiple choices of an encode-x strategy.

3.3 The Blocks World studies by Gray and colleagues

Gray et al. (2006) conducted three studies with the Blocks World task. In each experiment the difficulty of accessing the information was varied between subjects. The results of all three experiments indicate that the ease with which information can be accessed influences behavior in the Blocks World task. We will only discuss one of the experiments, as this is the one that was also modeled using ACT-R and using an ideal performer model (see Chapters 4, 6 and 7).

For a total of 48 trials, subjects had to copy a pattern of 8 blocks from the target window in the workspace window. In each trial the position of the blocks and their color was chosen randomly out of 16 positions and 8 colors. The interface of the experiment was as described in the beginning of this chapter: the windows that are depicted in Figure 3.1 were covered by gray rectangles. The information in the target window was revealed after a lockout time had passed.

The first ten trials were considered practice. The results show that lockout time has an effect for the measure of the number of blocks placed after the first visit to the target window. The resulting data is plotted in Figure 3.3 (lines between the findings of each condition are drawn for better understanding). As the lockout time increases, subjects tend to place (and thus study) more blocks after their first visit to the target window. The upcoming modeling sections will try to give an explanation how the lockout time can influence human behavior.

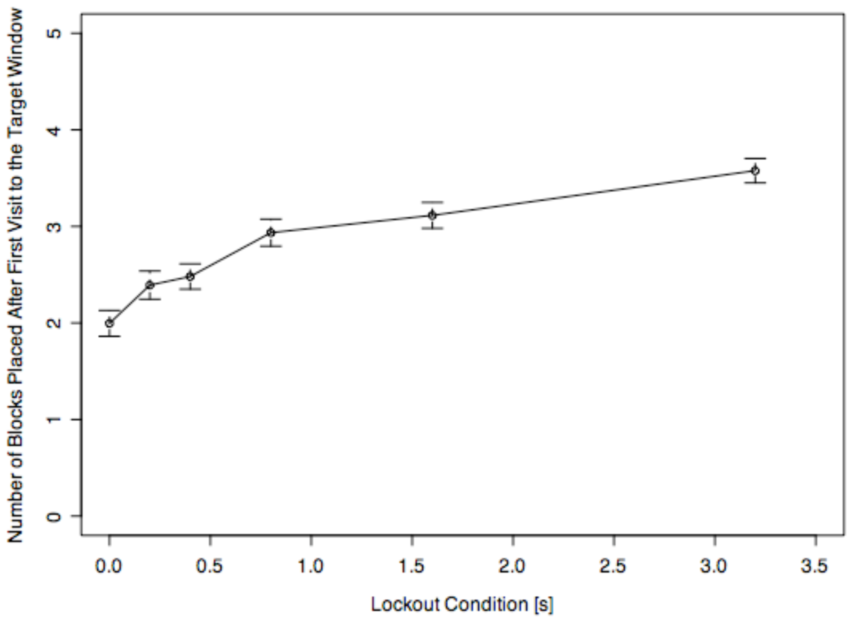

Figure 3.3: Average number of blocks placed after the first visit to the target window for each lockout condition (Gray et al., 2006). Lines between the findings of each condition are drawn for ease of comparison.

Chapter 4 The ideal performer model of the Blocks World task: a computational model that approximates human performance

The first model of the Blocks World task that we will discuss is an ideal performer model. Recall from Chapter 2 that ideal performer models try to capture what the optimal performance is in a given task, given the side conditions (Simon, 1992) that put constraints on the performance, and a criterion for which the model should optimize (Gray et al., 2006).

4.1 Structure of the model

Different side conditions occur at different steps of pursuing the goal to solve a trial of the Blocks World task (see Table 5 in Gray et al., 2006). Although each of these steps is related to basic perceptual, cognitive and motor operators for performing the task, there is no claim that the steps are executed exactly the same in the human mind. That being said, most steps have a direct mapping to (a combination of) the production rules of the ACT-R 5 models (see Chapter 6).

Each step that the model takes has a cost associated with it. This cost is the average time that the model (or a human) would spend on executing this step, expressed in milliseconds. The costs form the relevant side conditions that limit the model (and humans) to perform a trial of the Blocks World task as fast as possible. These time constraints have fixed values during each of the runs of the model, and most steps take an equal amount of time in each of the different experimental conditions. However, there are two steps that have variable costs. The exact magnitude that they have influences the total execution time of the model. As the model tries to minimize this time, these two steps influence what optimal behavior is in the task.

The first step that is of influence is the lockout time. The higher the lockout time is, the longer the model has to wait for each visit to the target window. The less visits it pays to the target window (by placing more blocks after each visit), the faster the trial can be completed.

The second step is related to memory. The more blocks a model studies per visit to the target window, the more time it will need per visit to the target window. As long as the model places every block that it studied, this is no problem. However, if the model forgets blocks, or almost forgets blocks, the time that was invested in studying the forgotten blocks is wasted.

4.2 Optimizing behavior

Given the side conditions, the model can learn what optimal behavior is. The ideal performer model of the Blocks World task learns this using the reinforcement learning algorithm Q-learning (Sutton & Barto, 1998; Watkins & Dayan, 1992). In Q-learning the usefulness, or utility, is learned for each action that is possible given a specific state of the world: a state-action pair. Q-learning relates rewards (or penalties) that it experiences in the world with the state-action pairs it used to get to the reward. Q-learning is a temporal difference learning algorithm (Sutton & Barto, 1998; Watkins & Dayan, 1992). This means that it relates state-actions less to a reward if they are further away from the moment where the reward is experienced (for equations and more details, read Gray et al., 2006).

The relevant state-action pairs in the Blocks World task model are the strategy choices. Strategies can be chosen at eight different states of the world. These states correspond to the number of blocks that the model has already placed in the target window (zero to seven). For each of these states a different number of actions is available, in the form of different encode-x strategies. For example, if the model has placed no block up until the strategy choice (at the beginning of the trial), it can use eight different strategies (encode-1 until encode-8). In contrast, when the model has already placed seven blocks, it can only use one strategy (encode-1). In total there are thus 36 state-action pairs (8+7+6+5+4+3+2+1).

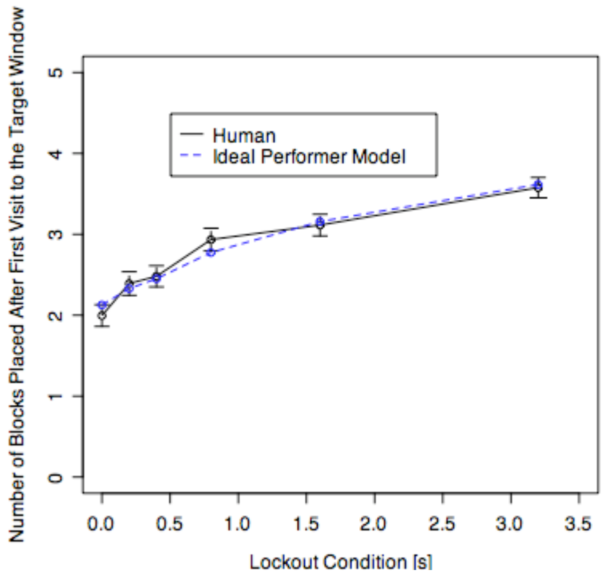

Figure 4.1: Performance of the ideal performer model compared with human performance for each of the lockout conditions (Gray et al., 2006). Lines between the findings of each condition are drawn for ease of comparison.

Rewards are given once per trial, at the end of a trial. The reward takes the form of a negative penalty that has the magnitude of the amount of time that is spend on the task since the mouse first entered the target window. Given enough experience, the model will learn to favor the use of state-action pairs that minimize the magnitude of the penalty (Gray et al., 2006).

4.3 Procedure

There were three phases in the development of the model: parameter exploration, training and testing. In the parameter exploration phase, the value of parameters that influence the memory retrieval time were explored in a grid search (see Gray et al., 2006, for details). The parameters were selected to optimize the measure of the number of blocks that the model places after the first visit to the target window.

Once the parameters were set, the training phase started. During this phase, six models were run for 100.000 trials in each of the experimental lockout conditions. In each trial, the state-action pairs that were used were chosen randomly. The utility values of the state-action pairs were then updated using the Q-learning algorithm. As a high number of training trials was run, and actions were randomly chosen, the model gained experience with all possible state-action pairs (in fact it has been shown that an extensive training like this can help the model find the optimal solution Sutton & Barto, 1998; Watkins & Dayan, 1992).

After the training phase there were six different models, each optimized to a specific lockout condition, and with specific utility values for each state action pair. Based on these utility values and a probability of retrieving the number of studied blocks (calculated using the base-level equation, Anderson, 2007; Anderson et al., 2004; Anderson & Schooler, 1991), one can determine how many blocks an ideal performer model would study and place on average during its first visit to the target window on a new trial (Gray et al., 2006).

4.4 Results

The models performance is plotted in Figure 4.1. It provides a good fit to the human data with a RMSE of 0.092 and an R^2 of 0.969 (Gray et al., 2006).

4.5 Discussion of model performance

The ideal performer model was modeled in such a way that it optimizes performance for the total amount of time it spends on the task. As it provides a close fit to the human data, it seems to support the idea that humans also optimize their performance in terms of the amount of time they spend on the task. Humans and model do not just study and place a small set of blocks, during each visit to the target window, as is predicted by the minimum memory hypothesis. Rather, the amount of blocks they study depends on the lockout condition. If interaction is made more costly, the number of blocks that is studied increases.

Despite the good fit of the model, it is still a good effort to develop a cognitive model of the Blocks World task. For one thing, an ideal performer model is not an exact model of human cognition (Gray et al., 2006). It can therefore not provide insight in the cognitive processes that the task involves at the level of detail that a cognitive model can give. If an ACT-R model is successful in modeling human performance in a task, it can thus give more insight in the cognitive processes involved in this task.

The fact that a lot of the structure and side-conditions (and their costs) of the ideal performer model are based on the ACT-R 5 models of the Blocks World task (Gray et al., 2005, see also Chapter 6) might give the impression that an ACT-R 6 model of the task should be easy to develop. However, a direct mapping from ideal performer model to ACT-R 6 models is not as easy as it seems. This has several reasons. Most importantly, the ideal performer model has gained its experience by training during 100.000 trials. Such a large set is required to learn the utility of all available state-action pairs. Humans have no experience with the Blocks World task before they participate in the experiment. It seems unlogical to incorporate this prior experience into an ACT-R model. This model's primary experience should be the same as the primary experience of humans, and the ACT-R model should be able to explain how these (initial) preferences are learned.

The ideal performer model incorporates thirty-two state-action pairs to incorporate eight generic encode-x strategies. The Q-learning algorithm requires this form of specifying state-action pairs. However, it is not necessarily compatible with the way in which humans choose their strategies. The success of specific encode-x strategies is not bound to specific states of the world. Some intuitive examples will illustrate this point. First of all, if a low encode-x strategy such as encode-2, was successful after the first visit to the target window, then humans might keep on using this strategy in successive visits. Similarly, if one experienced that a high encode-x strategy such as encode-6 is not successful (i.e., does not result in the placing of all six blocks) at the beginning of a trial, one might infer that this strategy will also be unsuccessful during successive visits to the target window.

The ideal performer model can be used as a source of inspiration for developing the cognitive models. Especially, the ideal performance models show that reinforcement learning based methods can be helpful in approximating human behavior in the Blocks World task. It also showed that it is useful to give the model feedback on performance using a single reward at the end of a trial. Also, it seems good to let this reward have the magnitude of the amount of time spend on the task. We will thus include a model with this type of reward function in our ACT-R 6 models. If the ACT-R models are also successful in modeling performance in the Blocks World task, they can give the necessary additional insight in the cognitive processes that are involved in the task.

Chapter 5 The ACT-R theory of cognition

In the next two chapters we will discuss several models of the Blocks World task that have been developed using the cognitive architecture ACT-R (Anderson, 2007; Anderson et al., 2004). ACT-R is a theory of human cognition that has been specified in a computational framework. Within this framework models of specific phenomena and tasks can be developed and tested. The theory of ACT-R has been adapted quite some times since its first introduction (see Anderson, 2007, for some discussion on this topic). With the changes of the theories about the structure of the architecture, also came new versions of the computational framework. The present study will compare performance of a model of the Blocks World task in the latest version of ACT-R, ACT-R 6 (Anderson, 2007; Bothell, 2005), and performance of a model that was developed for the same task in the previous version of ACT-R, ACT-R 5 (Anderson & Lebiere, 1998). Both versions differ in the way in which strategy preferences can be learned. Before discussing these and other differences between the two versions, we will first discuss the common aspects that form the core of the architecture.

5.1 General structure of the ACT-R cognitive architecture

ACT-R has a modular structure (Anderson, 2007). Each module acts relatively independent of other modules, and is specialized in processing data of a specific type. All of the modules can process relevant information in parallel. However, when the model is running, it can only access a subset of this information at each point in time, formally one chunk that is placed in a *buffer*. For example, the vision module can process the whole visual scene, but the model can only pay attention to at most one visual object at a time. For more detail we refer readers to (Anderson, 2007; Anderson et al., 2004).

Modules can determine the content of buffers, but this is also influenced by a central production system. This production system contains a set of production rules, or condition-action pairs, that take the form of if-then rules (Anderson, 2007; Anderson et al., 2004). Production rules have an if-side (or left-hand side) and a then-side (or right-hand side). On the if-side of the rule some checks are being made for the contents and current states of the buffers (for example, if specific buffers are not busy). On the then-side, some buffer modifications are made (for example, a motor movement is initiated). As soon as the contents and current states of the buffers match with the checks on the left-hand-side of a production rule this rule can execute, or *fire*.

During a process called conflict resolution, the model determines which production rule will fire. At first, the relevant production rules are selected by checking if the content and state of the buffers that are checked in the production rule (on the left-hand side) match the current content and state of the buffers. If more than one production rule is able to fire, the model decides which rule it fires based on a utility value that is associated with each individual rule. This utility value is a mathematical representation of previously experienced usefulness of the rule in achieving the overall goal. The higher the utility value, the more useful the rule was. If multiple production rules can fire, the one with the highest utility is chosen. The utility of production rules thus determines which production rules fire, and therefore which buffers are modified. These buffer modifications can influence which production rules can fire at a next conflict resolution round. Utility therefore influences behavior on the long run. We will also attribute strategy selection to the way these utilities are learned. If a specific strategy is incorporated in a (set of) production rules, then utility values can influence what strategy is used and it can influence the resulting behavior. The process by which utility is learned is called conditioning (Anderson, 2007).

ACT-R 5 and ACT-R 6 incorporate different methods for conditioning, and we will now discuss these.

5.2 Conditioning in ACT-R 5: probability learning

In ACT-R 5 conditioning was incorporated by a mechanism called probability learning (Anderson & Lebiere, 1998). In probability learning, it is hypothesized that people try to optimize the expected gain of each production rule that they use. This is calculated as follows (Lovett, 1998):

$$E_i = P_i G_i - C_i \qquad \text{(Expected gain equation)}$$

In this formula, the expected gain *E* of using production rule *i* is determined by the Cost *C* of executing this rule, the value of the goal that this production rule is achieving, *G*, and the estimated probability *P* of achieving a goal. The goal value is set as a parameter for each model, and is thought of as the amount of time one is willing to pursue a goal. This value (in most cases) does not alter during the learning process. The Cost of a production is estimated by the average time it takes after rule *i* has fired before a success or failure is encountered. The estimated probability of achieving a goal, P, is determined by the probability *q* that production rule *i* will achieve its intended next state (i.e., that a new production rule can fire) multiplied with the probability *r* that the production rule's goal can be reached after the rule has fired (Lovett, 1998).

In the case of the Blocks World task production rules will always achieve a next state, and as a result the value of *q* is always equal to 1. This leaves us with the parameter r. This parameter is calculated as the number of successes (both prior and experienced successes) divided by the number of successes and failures (both prior and experienced). As a model gains more experience with each production, the value of r diverges from its initial prior values, and converges towards the specific situation of the world that it faces. Production rules that have previously led to more successes and less failures relative to competing production rules, will have a higher value for r, and thus also for *P* in the expected gain equation (Lovett, 1998).

5.3 Problems identified with the probability learning mechanism of ACT-R 5

The probability learning mechanism was successfully used in ACT-R models to explain different sets of human behavior (e.g., Lovett, 1998). However, it also posed some challenges. Lovett (1998) showed that the probability learning mechanism can only slowly adapt to changes in the environment. The value of *r* (in the calculation of P) is determined by both prior and experienced successes and failures. Its value updates after each experience. However, the more experience the model has, the less influence a single event has on the value of r (Lovett, 1998). Behavior that has evolved due to a long range of experience (or due to strong prior values set by the modeler) can only be changed to new situations if the new situation persists for a long time.

More recently, the probability learning mechanism of ACT-R has been reexamined by Fu & Anderson (2004; 2006). They identified two problems. The first problem is that the system is limited to learning from binary feedback (e.g., there is either success or failure). This is in contrast with the real world, in which feedback can be more varied. Indeed, experiments have shown that people's behavior tends to be sensitive to the magnitude of the reward that they receive (for a short discussion, see Fu & Anderson, 2004).

The second issue that Fu and Anderson (2004; 2006) raised is related to the first one: given that success and failure is binary, and not scalar, how do you determine what a success is and what a failure is? Strategies that lead to partial success can only be labeled as complete success or complete failure, and not something in between.

5.4 Conditioning in ACT-R 6: utility learning based on reinforcement learning techniques

To overcome some of the issues with the probability learning mechanism, it was replaced with a reinforcement learning based mechanism in ACT-R 6 (Anderson, 2007; Fu & Anderson, 2004, 2006). Theoretically, this approach should solve some of the problems associated with the probability learning mechanism, as we will discuss in the next section. In addition, using reinforcement learning is also in line with recent findings in neuroscience (e.g., Carter et al., 1998;

Holroyd & Coles, 2002; Schultz, Dayan, & Montague, 1997) and with a trend in algorithms for cognitive architectures (e.g., Fu & Anderson, 2004; Nason & Laird, 2005; Sun, 1997).

The reinforcement learning algorithm that is implemented in ACT-R 6 is based on the temporal difference learning rule as described by Sutton and Barto (1998):

$$U(s_t)_{new} \leftarrow U(s_t)_{old} + \alpha\left[R(s_t) - U(s_t)_{old}\right] \qquad \text{(Temporal difference learning rule)}$$

The utility value *U* of a certain state *s* (or in case of ACT-R, a production) at time *t* will be updated when that state is used. Its new value, U_{new}, will be based on its old value, U_{old}, plus a *delta factor*. This delta factor is determined as the difference between the reward *R* that the model experienced at moment *t* and the previous estimate of the utility of that state that the model had, U_{old}. To make sure that learning is gradual, the delta factor is multiplied with a learning step α. This learning step limits the impact of the recent experience on the utility value. In regular reinforcement learning algorithms, the reward R is a combination of the immediate reward that a model receives by going to state s and an estimate of the rewards the model might get if it continues to take optimal actions from this state towards a certain goal. Reinforcement learning algorithms differ in how many future states they take into account for this estimate of the reward and in how heavily they weigh future rewards.

The temporal difference learning algorithm that is incorporated in ACT-R 6 differs from this general formula, in that it does *not* use estimates of future rewards to update the utility values of its production rules. Rather, it updates its utility values once a reward is given (this is the difference between a backward and forward view of reinforcement learning, see for example Sutton & Barto, 1998). At the moment in time when a reward is given, which we will denote as *j*, all production rules *i* that preceded the reward (and post ceded the previous reward) get their utility value updated as follows (Anderson, 2007):

$$U_i(n) \leftarrow U_i(n-1) + \alpha\left[\left(r_j - \Delta t_{i,j}\right) - U_i(n-1)\right] \qquad \text{(ACT-R's temporal difference rule)}$$

Note that the structure of this formula is very similar to the general temporal difference learning rule. Only, now U_i is the utility of production rule *i*, *r* is the *experienced* (and not estimated) reward at time *j*, and Δt denotes the time interval between *j* and the time at which production rule *i* fired. The interval between the time that a production fired and the time at which a reward was received will depend in part on how long it takes to process a production rule. We did not manipulate this in this study.

5.5 Solved and remaining problems of probability learning and utility learning

We will now describe how the new mechanism of utility learning addresses the issues raised by Lovett (1998) and by Fu and Anderson (2004; 2006) about its predecessor, probability learning. The main issue that Lovett (1998) raised is that the probability learning mechanism cannot act fast upon sudden changes in the reward structure of the environment. Reinforcement learning overcomes this problem due to the continuous comparison of experienced rewards *r* with the previously estimated utility (U_t[n-1]). If the world has changed, and the new experienced reward r differs a lot from the previous estimated utility U_t[n-1], then the difference between r and U_t[n-1] will be taken directly into account in the calculation of the new estimate of the utility (see the temporal difference learning rule). As a result, the utility values will directly converge towards the reward that is currently experienced in the environment. The amount of change in utility that the new experience will give depends on the magnitude of the learning step α. If α is close to one, the new value will closely match the newly experienced reward. If α is close to zero, it will take several runs before the model has completely adapted its value.

Fu and Anderson (2004; 2006) raised the issue that the expected gain equation is limited to binary feedback. In utility learning in ACT-R 6, rewards are not limited to binary feedback; they can have different magnitudes. Hence, this issue is solved. The second issue that Fu and

Anderson (2004; 2006) raised is that modelers have to define when a model experiences a success and when it experiences a failure. ACT-R 6's answer to this issue is two-sided. On the one side, a model can be rewarded for a partial success with a scalar value, and the modeler does not have to make a decision if these partial successes are complete successes or complete failures. On the other side, there are no formal guidelines *when* rewards should be given in models and *how big* rewards should be. This problem is thus still open ended, and we will address it in our study.

5.6 Exploration and exploitation of behavior

Given that a model has calculated the expected gains (ACT-R 5) or utilities (ACT-R 6) of production rules, the following question rises: which production rule should be selected out of competing production rules? On the one hand a model can always choose the production rule that has the highest utility value. This is exploitation of behavior (e.g., Russell & Norvig, 1995; Sutton & Barto, 1998). However, pure exploitation of behavior might lead to local maxima. It might also lead to bad performance in adaptive environments: if the reward structure in the world all of a sudden changes, the model does not know how good the other alternatives are, as their outcomes have not been explored. It might therefore be good to occasionally explore the world a bit (exploration of behavior, e.g., Russell & Norvig, 1995; Sutton & Barto, 1998), while at the same time exploiting a known set of successful actions (this is the exploration-exploitation trade-off, for a good discussion see Sutton & Barto, 1998).

ACT-R (both ACT-R 5 and ACT-R 6) balances exploration and exploitation by adding noise to the utility values of production rules each time that they can fire. Each of the production rules that is competing to fire gets its own noise value added to it, and the production that is eventually chosen to fire is the one that has the highest value based on its normal utility or expected gain, with the noise added to it. Before a model starts out with a task, possible competing experiences will have an equal utility or expected gain of 0 (unless this was set differently by the modeler). During conflict resolution, the production rule that fires will be fully determined by the random noise that is added to the available production rules. In other words, there will be full exploration. Once the first rewards (or successes and failures) have been experienced, the utility or expected gain of production rules changes. If different production rules lead to very different outcomes (and distinct expected gains or utilities), the influence of the noise that is added to them will be almost insignificant. The model will then start to fully exploit its learned behavior. However, as the noise can be both positive and negative, sometimes the production rule with the highest utility will get some utility subtracted, while competing production rules may get some utility added. This will lead to occasional exploration of behavior, even in cases where the model has learned that a specific competing production rule might actually be better than the one that is used.

5.7 Other differences between ACT-R 5 and ACT-R 6

Besides the difference in the method for incorporating conditioning, there are some other differences between ACT-R 5 and ACT-R 6. Most importantly, ACT-R 5 did not include an imaginal module (e.g., Anderson et al., 2004; Anderson & Lebiere, 1998). As a result, there was no separate buffer in which problem information could be represented. In the ACT-R 5 models this problem state information was thus contained in the goal module and in chunks that could be retrieved by declarative memory (Gray et al., 2005). We changed this for the ACT-R 6 model, as the imaginal module seemed a more natural place for this information.

Chapter 6: ACT-R 5 models of the Blocks World task

In this section we will outline the ACT-R 5 models of the Blocks World task. We will only discuss three out of the five developed models: the two models that were developed without altering the settings of ACT-R 5, and the best fitting model that was developed with alterations in the architecture. More details can be found in (Gray et al., 2005).

6.1 General structure of the ACT-R 5 models

The ACT-R 5 models of the Blocks World task incorporate the eight encode-x strategies (see Chapter 3) in the form of eight production rules. Each time that the model selects an encode-x strategy it is guided by the expected gain value of each of the strategies. In addition it is also guided by environmental constraints: an encode-x strategy can only fire if the model has placed at most eight minus x blocks.

Five different models were tested for their performance in the Blocks World task (Gray et al., 2005). The models differ in the moment when the outcome (success or failure) of their actions is experienced. This is either (a) once*,* at the end of the trial (a *once-weighted* model), or (b) each time that the model has placed a (set of) block(s) in the workspace window and either stops the trial by moving the mouse to the stop button or restarts studying blocks in the target window (an *each-weighted* model).

Both types of models were first tested without changing other aspects of the ACT-R architecture. We will refer to these models as *Vanilla* models. These models provided bad fits to the human data. This was ascribed to the way the expected gain was calculated. To test this hypothesis, the mechanism for calculating the expected gain was altered in three models. We will discuss the results of the best fitting model, the ACT-R 5 Each-Mixed-Weighted model.

6.2 General procedure for testing the models' performance

All ACT-R 5 models are equal except for the way that the outcome is given and the way the expected gain is calculated. The models interact with the same task interface as the human participants of the Blocks World task (see Chapter 3). Each model has been run in three different experimental lockout conditions: 0, 400 and 3200 milliseconds lockout. The models performed 48 trials per run. Data is reported for the models' performance in trials 25 until 48 and averaged over six runs. Data is compared with the data of Gray et al. (2006) on the measure of number of blocks placed after the first visit (and before the second visit) to the target window. A graph of all model results, compared to the human data is plotted in Figure 6.1 (lines between the findings of each condition are drawn for ease of comparison). For each model we calculated the R^2 and RMSE between model and human performance by comparing the mean model and mean human performance in each experimental condition. In this Chapter and the next Chapter we define a good quantitative fit as one in which the R^2 is bigger than 0.9 and the RMSE is smaller than 0.5. A fair fit is one in which either the R^2 is between 0.8 and 0.9, or the RMSE is between 0.5 and 1. If this is not the case, we call the fit bad.

6.3 Performance of the ACT-R 5 models

6.3.1 Vanilla ACT-R 5 models

The first two ACT-R 5 models use the regular way for calculating the expected gain (Lovett, 1998, see also Chapter 5 of this thesis). The Vanilla-Once-Weighted model receives feedback on successes and failures only *once*, at the end of a trial. The Vanilla-Each-Weighted model receives feedback on successes and failures according to an each-weighting scheme. Thus,

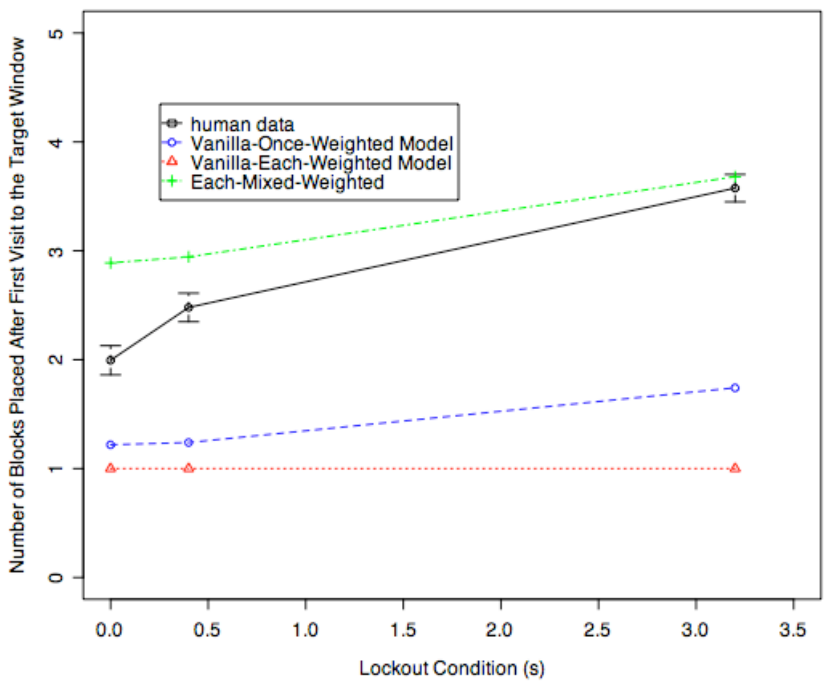

Figure 6.1: Results for the ACT-R 5 models in comparison with human performance data, per lockout condition (Based on Gray, Schoelles, & Sims, 2005). Lines between the findings of each condition are drawn for ease of comparison.

rewards are given each time that the model has tried to place a (set of) block(s) in the workspace window and either returns to the target window to study more blocks, or goes to the stop-button to stop the trial.

The performance of the models is depicted in Figure 6.1. Both models provide a bad quantitative fit to the human data. The RMSE of both models is bigger than 1.4; $R^2 = 0.33$ for the Vanilla-Once-Weighted model, and the R^2 of the Vanilla-Each-Weighted model cannot be calculated, as the model's behavior does not vary with variation in lockout condition. Both models also show a bad qualitative fit, as they undershoot human performance.

The explanation for the behavior of the model is due to the way credit is assigned in ACT-R 5. In both models, the goal value G was kept constant and is the same for each encode-x production. Also, both models had a constant value of P (the estimated probability of achieving a goal), as in the end they always reached a success (the once model always finished the trial, and the each-model always placed at least one block after each visit to the target window). Hence, the expected gain is determined by the value of the cost C of each production rule. Cost is determined by the time interval between the moment that a production rule fires and the moment that a success is experienced. Production rules that fire closer to a success marker get a lower cost. In case of our each model, the cost of a production rule is equal to the time between the moment an encode-x production rule fires and the time point when the model finishes placing a (set of) block(s) in the workspaces window (when a success or failure marker is encountered). The more blocks a model encodes (i.e., by choosing a higher encode-x strategy), the longer it will be occupied with studying blocks and with placing blocks. Hence, the higher the costs of the initial encode-x strategy will be. The strategy that encodes the least blocks, encode-1, will have the lowest costs. This production rule then has the highest expected gain. As a result, the model consistently places only one block at a time.

In the once-weighted model, a success marker is only encountered once at the end of the trial. Production rules that fire later in a trial (and closer to the success marker) have a lower cost than production rules that fire early in a trial. Higher encode-x production rules can only fire early on in the trial due to environmental constraints. They are always far away from the success marker, and therefore have a high cost. At the same time, models that try to remember a high number of blocks, might not always succeed in placing all of them, due to retrieval errors. They will still need lower encode-x strategies (e.g., encode-1 and encode-2) to place blocks. These lower encode-x strategies will thus be closer to the moment when the success is experienced, and receive a relative low cost. Even in situations where the model sequentially fires eight encode-1 strategies, is the average cost of these production rules (averaged over all uses) lower than those of competing encode-x production rules. The model uses both encode-1 and encode-2 rules, as both can fire relatively near to the goal.

6.3.2 Adapted ACT-R models

In three additional models, it was tested whether adaptations of the mechanism for calculating the expected gain of a production rule can give the ACT-R 5 models a better fit to the human data. Each of these additional models uses altered versions of the expected gain equation, to incorporate rewards that can have varying magnitudes. This way, they overcome the problem that has been identified in ACT-R 5 that it is limited to binary feedback. In each of these new models, instead of counting the amount of markers of success and failure that were encountered, the models count the *values* of these successes and failures. The value of a success is the amount of blocks that is correctly placed after a visit to the target window. The value of failures is the amount of blocks that is forgotten (i.e., studied but not placed). These values are then used in the equations for calculating the cost C and the probability of success P of using a production rule. The details of these formula's can be found in (Gray et al., 2005). For now it suffices to know that rewards are no longer limited to binary feedback.

The model's performance of the best fitting ACT-R 5 model, the Each-Mixed-Weighted model, is depicted in Figure 6.1. The model provides a good quantitative fit to the human data with $R^2 = 0.92$ and RMSE = 0.44. The model also provides a rather good qualitative fit, as it captures the trend that the number of blocks that the model places increases with an increase in lockout condition. Although the performance of the model does not yet have an exact fit with the human data, it does have the closest fit of all ACT-R 5 models. It thus seems favorable to have a model that can get scalar feedback on its performance.

6.4 General Discussion of the ACT-R 5 models of the Blocks World task

The results of these three models indicate that normal Vanilla ACT-R 5 models cannot capture the results of the human data in the Blocks World task (Gray et al., 2005). If the architecture is altered to distinguish between successes and failures of different magnitudes, the model fit improves drastically. In ACT-R 6 the use of scalar rewards does not require changes in the core mechanism, as ACT-R 6 uses reinforcement learning techniques (Anderson, 2007). Scalar rewards are common to use in reinforcement learning algorithms (e.g., Russell & Norvig, 1995; Sutton & Barto, 1998). It will thus be interesting to see if the ACT-R 6 models provide a better fit to the human data without specific "tweaking" of the mechanism.

Throughout this chapter we have assumed that the bad fits with the human data that the ACT-R 5 models provide are a result of flaws in the expected gain equation of ACT-R 5. However, alternative explanations are possible (see also Appendix A8 for some of our recommendations on alternative versions for modeling the Blocks World task). Most importantly, the set of production rules that is used to model the task could be wrong. To test this hypothesis, simulated versions of the ACT-R 5 Vanilla-Once-Weighted and Vanilla-Each-Weighted models were developed (Gray et al., 2005). These models did not include production rules and ACT-R parameters. They were run based on parameters and events that were measured in human data. The only aspect that was related to the ACT-R model was the way in which the simulated model learned to favor specific strategies. This was learned using the regular expected gain equation. The resulting model had about the same fit as the ACT-R 5 models from which it was derived. It thus seems that the expected gain equation has an impact on the model's behavior.

Chapter 7: ACT-R 6 models of the Blocks World task

In this chapter we describe the results of our model of the Blocks World task in ACT-R 6. Modeling human performance on this task in ACT-R 5 proved unsuccessful. The ACT-R 5 results indicate that using rewards of different magnitudes instead of binary rewards might improve the model's fit to human performance (Gray et al., 2005). Moreover, the study indicates that the mechanism by which the expected gain is calculated has the biggest influence on the fit; the structure of the production rules and the parameter values of ACT-R were of less influence (Gray et al., 2005).

ACT-R 6 uses temporal difference learning for calculating the utility of production rules. This mechanism can incorporate rewards of different magnitudes. We thus hypothesize that the ACT-R 6 models can provide a better qualitative and quantitative fit to the human data than ACT-R 5 models. In this Chapter we will first outline the general structure of the ACT-R 6 models. Then we give an overview of the different versions of the model that we test. For each model we will discuss the model's structure and the results.

7.1 General structure of the ACT-R 6 models

We will discuss four aspects of the general structure of the ACT-R 6 models: the chunks that are used, the production rule structure, the ACT-R parameters, and the reward structure. These issues are discussed briefly. For more details on each of these issues we would like to refer the reader to Appendix A2 and A3.

7.1.1 Chunks

Every object that an ACT-R 6 model encounters in the world is encoded as a chunk in declarative memory (Bothell, 2005). Hence, every visual object in the Blocks World task is represented as a chunk. It is the nature of the Blocks World task that not all visual objects are visible all the time: for each window (target, resource or workspace) either the gray cover-up of the window or the information that is underneath the cover-up is visible. Like humans, the ACT-R model can only attend to objects that are currently visible.

The model uses episodic memory chunks (Tulving, 1972) to store chunks in memory. Due to this, chunks that represent objects in different trials are unique. If chunks from a specific trial are retrieved from memory, they can not interfere with chunks that were made during previous trials. A more detailed discussion on our choice for episodic memory can be found in Appendix A2.

The model stores information about the current task in the goal buffer and the imaginal buffer. The goal buffer contains general information about the task. Most importantly it keeps track of the trial number and the sub task that the model currently performs. The imaginal buffer contains a representation of the problem at hand: what encode-x strategy has the model chosen, how many blocks has it studied, and what is the location of the studied blocks (in terms of row and column) in the target window.

7.1.2 Structure of the production rules

The behavior of the ACT-R 6 model is incorporated in production rules. More information about the structure of the production rules and flow charts of the production rules in different sub tasks can be found in Appendix A3. Crucially, two structurally different models are developed. Each of these models has a different set of production rules to perform the sub task of studying one or more blocks in the target window. The difference between the models is schematically illustrated in Figure 7.1.

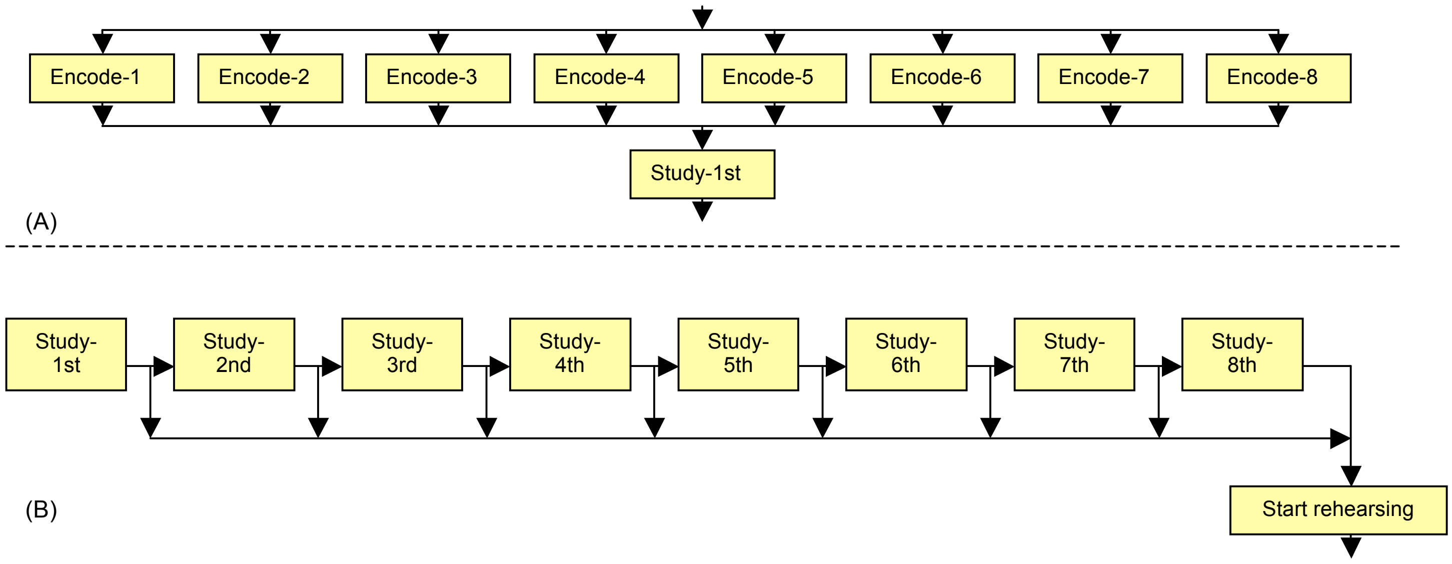

Figure 7.1: The structure of the critical production rule structure that determines the strategy in (a) The explicit model and (b) The implicit model.

The first model (depicted in Figure 7.1.A) makes explicit choices about the strategy that it uses. It is similar to the structure of the ACT-R 5 models (Chapter 6) and the ideal performer model (Chapter 4). Each time before it moves the mouse cursor to the target window (to uncover the information in the target window), it first decides how many blocks it wants to study during that visit to the target window. This strategy choice takes the form of an encode-x strategy, in which x is the number of blocks that will be studied in the target window during the upcoming visit. There are eight different encode-x strategies and each is incorporated as one production rule in the model. The model can learn to favor different production rules, and thus different strategies, using utility learning. We will refer to models that incorporate this set of production rules as “explicit models”, as there is one explicit choice for an encode-x strategy during each visit.

The second model (depicted in Figure 7.1.B) does not select one specific production rule that determines the entire strategy. Rather, this model always studies one block in the target window. It then decides if it should study a second block or if it should start rehearsing the blocks it learned so far. If it chooses to study the second block, it will study this block, and then decide if it should study a third block or start rehearsing the studied blocks. The model contains seven of these “study-yth block” production rules. “y” is the number of the block that is studied during this visit to the target window. Each of these production rules can fire once the model has studied y-1 blocks.

There is only one production rule that can start the rehearsal sub task (see Figure 7.1.B). Each time that the model has to decide whether to study an extra block in the target window, this production rule is in competition with one of the “study-yth block” production rules. If the model eventually decides to study eight blocks during one visit to the target window, then the “start-rehearsing” production rule has been in competition seven times. The model makes no single explicit choice to use a specific strategy. It determines its strategy implicitly by making multiple moment-by-moment decisions whether it should study an extra block or not. We will therefore refer to models that incorporate this set of production rules as “implicit models”.

The implicit model selects production rules based on their utility value. Note that the decision to study at least y blocks will always depend on the utility of the decision to study at least y-1 blocks. Once the model has learned that studying y blocks is unsuccessful (i.e., it cannot retrieve all y blocks), it will not attempt to study more than y blocks on any successive trial. This is in contrast with the explicit models, in which there is no explicit interdependency between the utility values of different encode-x strategies.

7.1.3 Parameter settings

We want our model to be a test of the utility learning mechanism of ACT-R. We thus keep most of the models parameters at their default values (documented in Bothell, 2008). We will mention some of the special parameters and the parameters that have a different value than the default. We have set the parameter settings for ACT-R's utility learning mechanism at their default values. Exception to this is the noise parameter of the utility learning mechanism, which is set to 1.5. In pilot runs of our model, this value turned out to offer a good balance between exploration and exploitation of the models' behavior.

In our exploration of parameter values we found that it is also sufficient to keep most of the parameters for declarative memory at their default values. With these basic settings, the maximum number of blocks that the model is able to retrieve after a normal round of study lies around three to four, which is close to human values. In rare cases, the model is able to retrieve up to eight studied blocks, just like humans. Two parameters for declarative memory are set to different values than the default ACT-R values. First, noise is added to the activation value of the chunks (with s set to 0.25). Second, our model does not use the optimized learning version of the base-level activation equation (for the difference between the optimized and the non-optimized equation, see Bothell, 2008). A study has shown that using this approximation can alter model performance (Sims & Gray, 2004).

7.1.4 Reward structure

The most important manipulation that we apply to our ACT-R models is the reward structure. In different models we manipulate two aspects of the reward structure: the moment when the reward is given and the magnitude of the reward. The moment when the reward is given is varied in the same manner as is done in the ACT-R 5 models of the Blocks World task (Gray et al., 2005). Rewards are given either (a) once at the end of a trial or (b) each time that the model has placed one or more blocks in the workspace window and either stops the trial (by moving the mouse to the stop button) or goes to the target window to start studying new blocks. We call the first type of models once-weighted models, as a reward is given once per trial. We call the second type each-weighted models, as a reward is given each time that part of the task has been completed.

The magnitude of the rewards is varied in two fundamental ways that relate to what the model (and the modeler) conceptualizes as a reward. On the one hand, a reward can be expressed in terms of *how good,* or how accurate, the model performs the task itself. For

Model-name	**When is reward given**	**Reward concept**	**Magnitude of reward**
Once-Fixed-Value-Weighted	Once	Accuracy	8 (= total # blocks placed)
Once-Total-Trial-Time-Weighted	Once	Time	-1 * total trial time
Once-Sum-Of-Lockout-Durations-Weighted	Once	Time	-1 * lockout size * # visits to target window
Each-Success-Weighted	Each	Accuracy	# blocks placed
Each-Success-Failure-Weighted	Each	Accuracy	# blocks placed – # blocks forgotten
Each-Blocks-Per-Second-Weighted	Each	Time	# blocks placed / time between moving mouse to target window and placing studied blocks

Table 7.1: The six different reward schemas used by both explicit and implicit ACT-R 6 models (# denotes "Number of").

example, how many blocks does it place after a visit to the target window, and how many blocks does it study but forget? A totally different conception of the reward structure is to express rewards in terms of *how fast* the model performs the task, or specific parts of it. Different combinations of when the rewards are experienced and reward concepts (or magnitudes) can lead to different weighting schemes. We use six reward schemes, which are summarized in Table 7.1. For each of these reward schemes we have indicated its name, the moment when the reward is given (once or each), the concept of the reward (accuracy or time) and the formula for calculating the magnitude of the reward. We will discuss the details of the formulas for calculating the magnitude of the reward in the sections that describe the corresponding models. Each of the six weighting schemes is used in both an explicit and an implicit model. This makes the total number of tested models twelve.

7.2 General Procedure for Testing the Models' Performances

All twelve models have the same declarative knowledge and general ACT-R parameter settings. The structure of the production rules is different for the explicit and implicit models. The models interact with the same task interface as the human participants of the Blocks World task (see Chapter 3 and Figure 3.1). We tested the models' performances in three different experimental lockout conditions: 0, 400 and 3200 milliseconds lockout. These are the same conditions in which the performance of the ACT-R 5 models are tested (Gray et al., 2005). For each experimental condition, each of the models was run six times (making the total number of runs per model eighteen). In each of these runs, the models performed 48 trials. We consider the first twenty-four trials as practice trials during which the model could settle in on a strategy (as is also done in the ACT-R 5 models, Gray et al., 2005). Note that humans settle in faster on a strategy.

We compare performance of the model with the human data on the measure of the number of blocks placed after the first visit to the target window. Per model, the data is averaged over all runs for trials 25 to 48. Human data is averaged over trials 11 to 48. The resulting data is visualized in Figure 7.2 for the explicit models and in Figure 7.3 for the implicit models.

To test the quantitative and qualitative fit of the model to the human data we calculated the R^2, RMSE and SSEs (sum of squared errors) between the average model performance and the average human performance. We define a good quantitative fit as one in which R^2 is bigger than 0.9, RMSE is smaller than 0.5 and the SSE for each of the three lockout conditions is less than 1. If two measures are good, but R^2 is between 0.8 and 0.9, RMSE is between 0.5 and 1 or one of the SSE is bigger than 1, then we call the fit fair. If two or more measures are not good, we call the fit bad.

For each of our models we test our hypothesis that ACT-R 6 models of the Blocks World task provide a better qualitative and quantitative fit to the human data than ACT-R 5 models of the task. In order to do so, we compare the calculated measures of fit with those of the best fitting ACT-R 5 model: the ACT-R 5 Each-Mixed-Weighted Model. This model has an R^2 of 0.92 and a RMSE of 0.44. As this model required changes in the ACT-R architecture, we will also compare our model results to those of the ACT-R 5 Vanilla models, which do not include any changes to the architecture. These models have RMSEs of 1.46 (Vanilla-Once) and 1.87 (Vanilla-Each). The Vanilla-Once-Weighted model has an R^2 of 0.33. The R^2 of the Vanilla-Each-Weighted model cannot be calculated, as behavior does not vary with the lockout condition.

7.3 Performance of the ACT-R 6 models

7.3.1 ACT-R 6 Explicit Once-Weighted models

7.3.1.1 ACT-R 6 Explicit-Once-Fixed-Value-Weighted

Model structure

The model uses the explicit strategy choice structure. Rewards are given once at the end of the trial. The concept of the reward is accuracy: how many blocks does the model place? As the model has always placed eight blocks at the end of the trial, the reward has a fixed value of eight.

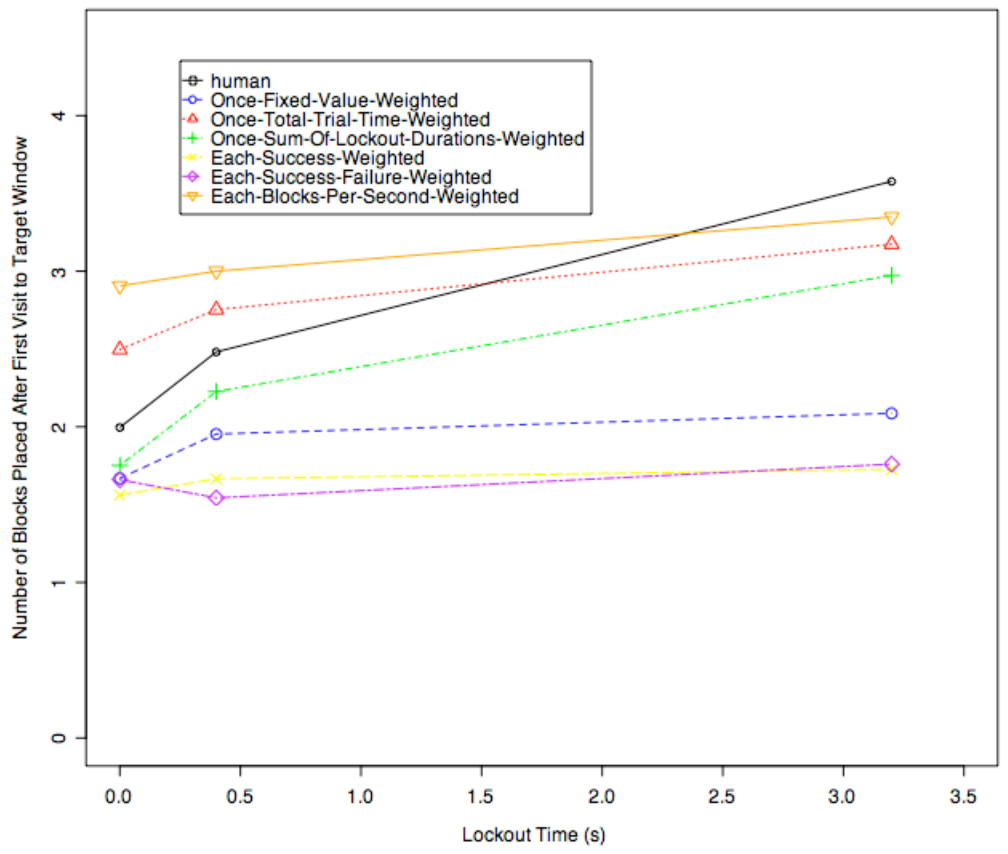

Figure 7.2: Average number of blocks placed after the first visit to the target window for the explicit models and the human data. Model data is averaged over six runs for trials 25 until 48. Lines between the findings of each condition are drawn for ease of comparison.

Results
The model's performance is plotted in Figure 7.2. The model provides a fair quantitative fit to the human data with R^2 = 0.994, RMSE = 0.930 and SSE of 0.11, 0.41 and 2.07 (for lockouts of respectively 0, 400 and 3200 milliseconds). The qualitative fit is less good (also visible in the box plots in Appendix A4). The model undershoots the data. It places one or two blocks in fifty percent of the trials of both the 0 and 400 milliseconds lockout condition. In the 3200 milliseconds condition it places up to three blocks.

Discussion of model performance and comparison with ACT-R 5 models
The quantitative fit for this model is about the same as the fit of the best ACT-R 5 model (R^2 of the current model are better; RMSE is worse). Qualitatively, the current model shows more variance in its performance (which is evident in the broader quartiles of the box plot in Appendix A4). We thus argue that the fit of the current model is worse than that of the best ACT-R 5 model. If we only look at the median of the model data, then the model seems to be behaving very similar to the ACT-R 5 Vanilla-Once-Weighted and the ACT-R 5 Vanilla-Each-Weighted models (the ACT-R 5 models with the worst fit): it always places around one or two blocks.

An explanation for the model's behavior can be found in ACT-R 6's utility learning mechanism. The model always receives a reward of eight, no matter which strategies are chosen. The difference between the utilities of different encode-x strategies will thus fully be determined by temporal discounting: the smaller the distance between the time the reward was given and the time that the encode-x production rule fired, the bigger the utility of this production rule. Low encode-x strategies will have relatively high utility values: it takes less time to complete these strategies, and they are more likely to fire near the end of a trial (and close to the reward) due to environmental constraints. Even if the model uses eight sequential encode-1 strategies, the average utility value will be low.

A contrasting example would be when the model uses two encode-4 strategies in succession. The first time that the encode-4 strategy fires will have less distance from the end of the trial than in the case of the encode-1 strategy. Hence, it will have a higher utility. However, the second time that the encode-4 production fires it is further away from the goal than the last time the encode-1 strategy in our other example fired. On average it turns out that these utility values are lower than those of the encode-1 strategies. In the end the encode-1 strategy gets the highest utility value in every lockout condition (see also the table with utility values in Appendix A6). The model also chooses other strategies during the complete set of trials 25-48, due to effects of noise and learning.

7.3.1.2 ACT-R 6 Explicit-Once-Total-Trial-Time-Weighted

Model structure

The model uses the explicit strategy choice structure. Rewards are given once at the end of the trial. The concept of the reward is time. The reward takes the magnitude of the total time needed to complete the trial times minus one. It thus forms a negative penalty.

Results

The model's performance is plotted in Figure 7.2. The model provides a good quantitative fit to the human data with $R^2 = 0.993$, RMSE = 0.396 and SSEs of 0.22, 0.06 and 0.20 (for lockouts of respectively 0, 400 and 3200 milliseconds). However, the qualitative fit is less good (which is also visible in the box plot in Appendix A4). The model overshoots the human data in the 0 and 400 milliseconds lockout condition, and undershoots the data in the 3200 milliseconds condition. However, the model correctly places more blocks per visit to the target window in the 3200 milliseconds lockout condition.

Discussion of model performance and comparison with ACT-R 5 models

The current model provides a better quantitative fit to the human data than the best ACT-R 5 model. Qualitatively, the current model performs worse than the ACT-R 5 model, as it has a broader variance (visible in the box plots).

The model correctly increases the number of blocks placed after a first visit to the target window if the lockout time increases. This is due to the reward structure: the more blocks the model places, the smaller the penalty (or the higher the reward) will be. As placing more blocks per visit to the target window in general decreases the amount of time spend on the task, the model places more blocks than the Explicit-Once-Fixed-Value-Weighted model. The model overshoots the human data in the 0 and 400 milliseconds lockout conditions. In contrast, in the 3200 millisecond condition it undershoots the data. In general the results for the different lockout condition are very similar to each other. An explanation for this is that the magnitude of the lockout condition is not that big compared to the total time spend on the task. Thus, the influence of the lockout time on the reward is not that big, and behavior is not altered that much between lockout conditions.

As can be seen in the box plots in Appendix A4, the model data varies a lot. The model also uses low encode-x strategies. This is again due to the fact that these strategies can fire close to the end of a trial (and thus close to the moment when the reward is experienced). Due to temporal discounting, these encode-x strategies will be favored more than high encode-x strategies. However, if the low encode-x strategies are used too much on a trial, then the magnitude of the reward will become very low, and the model will learn again to disfavor these strategies.

7.3.1.3 ACT-R 6 Explicit-Once-Sum-Of-Lockout-Durations-Weighted

Model structure
The model uses the explicit strategy choice structure. Rewards are given once at the end of the trial. The concept of the reward is time. In this case, not the total trial time is taken into account. Instead, the reward focuses on the total lockout time that the model experiences. The reward takes the magnitude of minus one times the magnitude of the lockout condition times the number of visits to the target window (i.e., the total time spend waiting for the lockout). It thus forms a penalty that increases with the lockout condition and with the number of visits to the target window.

Results
The model's performance is plotted in Figure 7.2. The model provides a good quantitative fit to the human data with $R^2 = 0.990$, RMSE = 0.434 and SSEs of 0.10, 0.09 and 0.38 (for lockouts of respectively 0, 400 and 3200 milliseconds). The model provides the best quantitative fit to the human data of all the ACT-R 6 models. The qualitative fit is not that good. This can be seen in the box plots in Appendix A4. As the lockout time increases, the model correctly uses higher encode-x strategies more often. This is reflected in higher median values for higher lockout conditions. Similar to the previous models, the model also uses lower encode-x strategies often at the beginning of the trials.

Discussion of model performance and comparison with ACT-R 5 models
The current model provides a better quantitative fit to the human data than the best ACT-R 5 model. It is debatable which model (the current ACT-R 6 model, or the best fitting ACT-R 5 model) provides a better qualitative fit. The median of the number of blocks placed after the first visit to the target window is one lower than the median of the human data for each of the three lockout conditions and for both models. The ACT-R 5 model has a relatively smaller variance than the ACT-R 6 model (compare the box plot of the ACT-R 5 Each-Mixed Weighted model in Appendix A1 with the box plot of the current model in Appendix A4). However, the ACT-R 6 model does a better job at capturing the trend of placing more blocks (after the first visit to the target window) as the lockout time increases.

As the reward of the model is related to the magnitude of the lockout condition and the number of visits to the target window, the model is reinforced extra to finish trials fast in higher lockout conditions (as each visit to the target window has the biggest impact on the reward magnitude in the highest lockout condition). It thus learns to favor higher encode-x strategies in higher lockout conditions. However, it still also favors low encode-x strategies (see for example the utility values in the utility table in Appendix A6). The explanation is the same as given before: lower encode-x production rules can fire close to the end of a trial and are favored by temporal discounting.

7.3.2 ACT-R 6 Explicit Each-Weighted models

7.3.2.1 ACT-R 6 Explicit-Each-Success-Weighted

Model structure
The model uses the explicit strategy choice structure. Rewards are given according to the each weighting scheme, as soon as part of the task is completed: after the model places a (set of) block(s) in the workspace window and either returns back to the target window to study more blocks or returns to the stop-button to stop the trial. The concept of the reward is accuracy. The magnitude of the reward is the number of blocks that the model has placed correctly in the workspace window after the last visit to the target window.

Results
The model's performance is plotted in Figure 7.2. The model provides a bad quantitative fit to the human data with $R^2 = 0.836$, RMSE = 1.183 and SSEs of 0.20, 0.64 and 3.58 (for lockouts of respectively 0, 400 and 3200 milliseconds). The qualitative fit is also bad. The model consistently

undershoots the human data, and uses the encode-1 or encode-2 strategy during most visits to the target window, independent of the lockout condition.

Discussion of model performance and comparison with ACT-R 5 models
The model provides a bad fit and performs as bad as the worst ACT-R 5 models. The explanation for its behavior is similar to the one given for the ACT-R 5 Vanilla-Each-Weighted model. A reward is given each time that the model has placed a set of blocks in the target window. Hence, there is exactly one reward after each use of an encode-x strategy. The magnitude of the reward varies depending on the model performance. The more blocks the model places, the higher the received reward is. However, the increase in reward between an encode-x and an encode-(x+1) rule is only one, while the temporal cost for studying and placing one block extra on average takes the model at least four seconds. As the temporal discounting algorithm extracts this time from the reward, the increase in utility of an associated encode-(x+1) production rule compared to an encode-x production rule is 1 - 4 = -3. As a result, the model mostly uses the strategies that have the lowest costs: the encode-1 and encode-2 strategies (other low strategies are sometimes selected due to noise).

7.3.2.2 ACT-R 6 Explicit-Each-Success-Failure-Weighted

Model structure
The model uses the explicit strategy choice structure. Rewards are given according to the each weighting scheme as soon as part of the task is completed: after the model places a (set of) block(s) in the workspace window and either returns back to the target window to study more blocks or returns to the stop-button to stop the trial. The concept of the reward is accuracy. The magnitude of the reward is the difference between the number of blocks that has been placed in the workspace window and the number of blocks that has been forgotten (i.e., studied but not placed) after the last visit to the target window. Thus, this model incorporates both the success of the model (how many blocks has it placed) and the failures (how many blocks has it studied but not placed).

Results
The model's performance is plotted in Figure 7.2. The model provides a bad quantitative fit to the human data with R^2 = 0.236, RMSE = 1.216 and SSE of 0.11, 0.85 and 3.58 (for lockouts of respectively 0, 400 and 3200 milliseconds). The qualitative fit is also bad. The model consistently undershoots the human data and uses encode-1 and encode-2 strategies during most visits to the target window, independent of the lockout condition.

Discussion of model performance and comparison with ACT-R 5 models
The conclusions for this model are the same as for the Explicit-Each-Success-Weighted model: there is a bad qualitative and quantitative fit to the human data and the fit is not better than any of the ACT-R 5 models. Rather, the fit is very similar to the ACT-R 5 Vanilla-Each-Weighted models. The explanation for the model's behavior is again that the temporal discounting algorithm favors recent choices too much.

7.3.2.3 ACT-R 6 Explicit-Each-Blocks-Per-Second-Weighted

Model structure
The model uses the explicit strategy choice structure. Rewards are given according to the each weighting scheme, as soon as a part of the task is completed: after the model places a (set of) block(s) in the workspace window and either returns back to the target window to study more blocks or returns to the stop-button to stop the trial. The concept of the reward is time. The formula for calculating the reward magnitude is rather complex when compared to the other rewards. Before explaining this formula, we will first explain why the regular alternatives for capturing time aspects in the reward (i.e., the ones that are used for the once-models) are impractical to use.

If the reward would take the magnitude of minus one times the amount of time spend on studying and placing blocks between two visits to the target window (equal to the weighting scheme of the Once-Total-Trial-Time-Weighted model), then the reward would be least negative if the model studied and placed only one block. This is far from what we observe in human participants and therefore not a good weighting scheme. If we would use a weighting scheme similar to the Once-Sum-Of-Lockout-Durations-Weighted Model, then the model would receive a reward that has the magnitude of the lockout condition. As the lockout time is always the same for a run in one lockout condition, the reward would have no different magnitude no matter what strategy the model uses. This is also unfavorable.

Therefore, the reward takes the magnitude of the number of blocks that is placed in the workspace window divided by the total time that passed between the time that the model moved the mouse to the target window to start the studying and the time at which it stopped placing blocks in the workspace window. In a sense, this is the number of blocks that the model places per second. The range of reward magnitudes in this model is very small when compared to those of other models (between 0 and about 2.5, instead of for example between 0 and 8 in the Each-Success-Weighted models). If we use temporal discounting on these rewards, their effect will be completely overshadowed by this mechanism. We thus decided to update the utilities of production rules without temporal discounting. Due to the small magnitude of the rewards we also had to decrease the magnitude of the utility noise to 0.1.

Results

The model's performance is plotted in Figure 7.2. The model provides a fair quantitative fit to the human data with $R^2 = 0.981$, RMSE = 0.636 and SSEs of 0.88, 0.28 and 0.05 (for lockouts of respectively 0, 400 and 3200 milliseconds). The qualitative fit is bad. As can be seen in the box plots in Appendix A4, the distribution of the selected strategies is about the same for each of the lockout conditions. The model does not learn to behave differently for each of the lockout conditions. As a result, the model overshoots the human data in the 0 and 400 milliseconds lockout condition, and undershoots the human data in the 3200 milliseconds lockout condition.

Discussion of model performance and comparison with ACT-R 5 models

The current model performs worse than the best performing ACT-R 5 model. It overshoots human performance in the 0 and 400 milliseconds lockout condition. Unlike the other ACT-R 6 Explicit-Each-Models, rewards in the current model are not discounted by the amount of time spend on the task. As a result, the model does not show an extreme favoring of low encode-x strategies. Rather, the model seems to optimize performance in time: it tries to place as much blocks as possible after each visit to the target window. The model experiences one lockout time, before each reward is given. The more blocks the model places, the smaller the impact of the lockout time on the reward magnitude. Thus, it favors to study and place more blocks during one visit to the target window.

However, as the model studies more blocks at a time, the activation levels of the blocks that were studied decrease (it takes time before the model places the last block it studied) and their values might go below the retrieval threshold. As a result, the model cannot retrieve all chunks and will spend too much time on studying blocks, without placing them. Hence, the received reward (and as a result, the utility of the associated encode-x production rule) will be lower.

As can be seen in Table 7.2, the utility values of encode-1 to encode-4 differ from each other, but are very close. Although the noise of the utility learning mechanism has been adjusted for this (lowered to 0.1), the model still does not exploit only one specific strategy, but chooses multiple to perform its actions. If the noise were decreased more, the model would have difficulty to explore the environment enough in the beginning of the training.

7.3.3 ACT-R 6 Implicit Once-Weighted models

7.3.3.1 ACT-R 6 Implicit-Once-Fixed-Value-Weighted

Model structure
The model uses the implicit strategy choice structure. Rewards are given once at the end of the trial. The concept of the reward is accuracy: how many blocks does the model place? As the model has always placed eight blocks at the end of the trial, the reward has a fixed value of eight.

Results
The model's performance is plotted in Figure 7.3. The model provides a bad quantitative fit to the human data with $R^2 = 0.056$, RMSE = 1.774 and SSEs of 0.85, 2.17 and 6.42 (for lockouts of respectively 0, 400 and 3200 milliseconds). The qualitative fit is also bad: the model consistently undershoots the human performance and places only one block after each visit to the target window, independent of the lockout condition.

Discussion of model performance and comparison with ACT-R 5 models
The current model performs worse than any of the ACT-R 5 models. Just like the ACT-R 5 Vanilla-Each-Weighted model, this model undershoots human performance, and only places one block per visit to the target window. This can again be explained in terms of the utility learning mechanism. The smaller the distance between the time the reward is received and the time that a production rule fired, the smaller the magnitude of the reward will be. During each of the trials, eventually the model will fire the production rule "start rehearsing", no matter how many blocks the model places. This production rule is in competition with the production rules that might increase the number of studied blocks ("study-yth block"). The production rule that starts the rehearsal always fires after the last study-yth block production fired, and is always closer to the goal than its competing production rules. As a result this production rule will always get the highest utility value. In future cases of conflict resolution, the start rehearsal production rule will always win the competition from any study-yth block production rule. Hence, the model will perform stable suboptimal behavior.

7.3.3.2 ACT-R 6 Implicit-Once-Total-Trial-Time-Weighted

Model structure
The model uses the implicit strategy choice structure. Rewards are given once at the end of the trial. The concept of the reward is time. The reward takes the magnitude of the total time needed to complete the trial times minus one. It thus forms a negative penalty.

Results
The model's performance is plotted in Figure 7.3. The model provides a bad quantitative fit to the human data with $R^2 = 0.894$, RMSE = 1.276 and SSEs of 0.87, 2.11 and 1.91 (for lockout of respectively 0, 400 and 3200 milliseconds). The qualitative fit is also bad. The model consistently places one block in the workspace window in the 0 and 400 milliseconds lockout condition. Only in the 3200 milliseconds lockout condition does the model place more blocks after the first visit to the target window.

Discussion of model performance and comparison with ACT-R 5 models
The model provides a bad fit to the human data, and performs worse than the best ACT-R 5 model: it undershoots the data. The fit is very similar to that of the ACT-R 5 Vanilla-Once-Weighted model. The explanation for the behavior of the current model is the same as given for the ACT-R 6 Implicit-Once-Fixed-Value-Weighted model: no matter how low the received reward is, the time at which the "start rehearsal" production rule is fired is always closest to the time at which the reward is given. As a result, this production rule will be penalized less than other production rules that caused the behavior (e.g., less than the "study-yth block" production rules). The model will only learn to place more than one block per visit to the target window when the lockout cost is severely high. In this case trial time (and also reward magnitude) increases

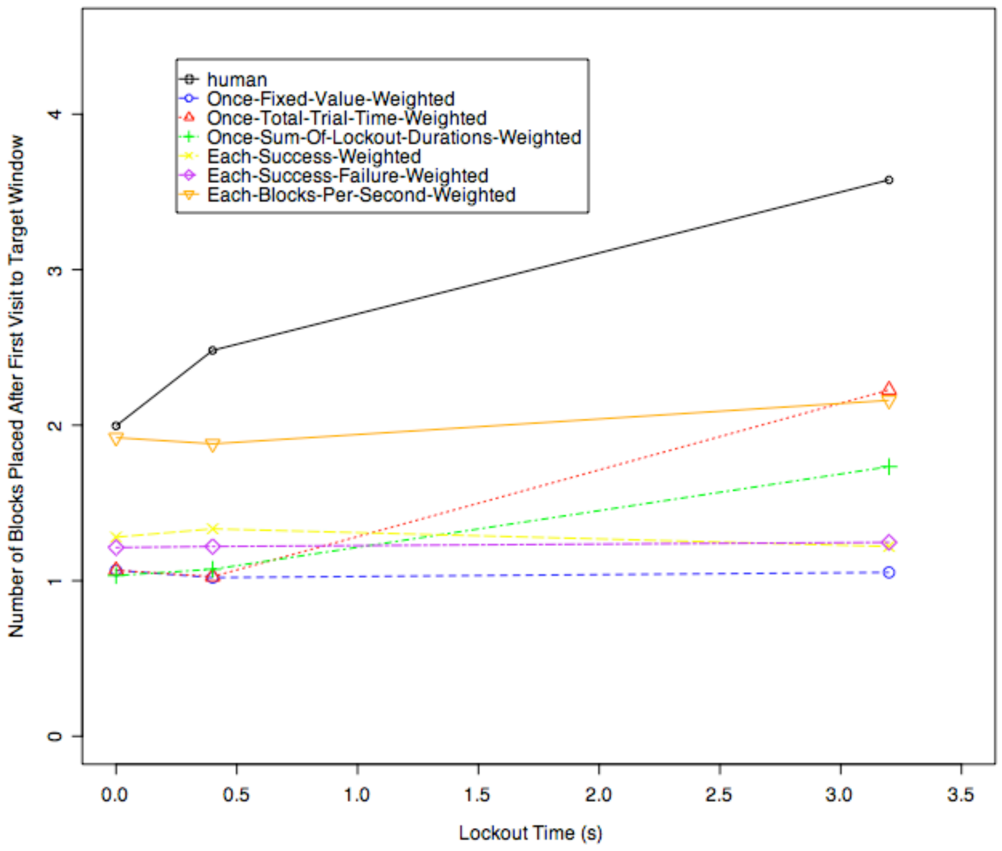

Figure 7.3: Average number of blocks placed after the first visit to the target window for the implicit models and the human data. Model data is averaged over six runs for trials 25 until 48. . Lines between the findings of each condition are drawn for ease of comparison.

strongly if only a few blocks are placed. However, even in this case the model will only study one block during most of the trials (about fifty percent of the trials, see the box plot in Appendix A5).

7.3.3.3 ACT-R 6 Implicit-Once-Sum-Of-Lockout-Durations-Weighted

Model structure
The model uses the implicit strategy choice structure. Rewards are given once at the end of the trial. The concept of the reward is time. In this case, not the total trial time is taken into account. Instead, the reward focuses on the total lockout time that the model experiences. The reward takes the magnitude of minus one times the magnitude of the lockout condition times the number of visits to the target window (i.e., the total time spend waiting for the lockout). It thus forms a penalty that increases with the lockout condition and with the number of visits to the target window.

Results
The model's performance is plotted in Figure 7.3. The model provides a bad quantitative fit to the human data with R^2 = 0.939, RMSE = 1.497 and SSEs of 0.92, 1.98 and 3.82 (for lockouts of respectively 0, 400 and 3200 milliseconds). The qualitative fit is also bad. The model undershoots the human data. It consistently places one block in the workspace window in the 0 and 400 millisecond lockout condition. In the 3200 millisecond lockout condition it places one block most of the time, but sometimes two.

Discussion of model performance and comparison with ACT-R 5 models
The model provides a bad fit to the human data, and it performs worse than the ACT-R 5 models: it undershoots the data. The fit is very similar to that of the ACT-R 5 Vanilla-Once-Weighted model. The explanation for this bad fit is the same as given for the previous ACT-R 6 Implicit-Once-Weighted models.

7.3.4 ACT-R 6 Implicit Each-Weighted models

7.3.4.1 ACT-R 6 Implicit-Each-Success-Weighted

Model structure
The model uses the implicit strategy choice structure. Rewards are given according to the each weighting scheme, as soon as part of the task is completed: after the model places a (set of) block(s) in the workspace window and either returns back to the target window to study more blocks or returns to the stop-button to stop the trial. The concept of the reward is accuracy. The magnitude of the reward is the number of blocks that the model has placed correctly in the workspace window after the last visit to the target window.

Results
The model's performance is plotted in Figure 7.3. The model provides a bad quantitative fit to the human data with $R^2 = 0.363$, RMSE = 1.568 and SSEs of 0.53, 1.35 and 5.5 (for lockouts of respectively 0, 400 and 3200 milliseconds). The qualitative fit is also bad. The model consistently undershoots the human data and places only one block after most of the visits to the target window, independent of the lockout condition.

Discussion of model performance and comparison with ACT-R 5 models
The model provides a bad fit to the human data and performs worse than most ACT-R 5 models: it undershoots the data. The fit is very similar to that of the ACT-R 5 Vanilla-Once-Weighted model. The explanation for this bad fit is the same as given for the ACT-R 6 Implicit-Once-Weighted models.

7.3.4.2 ACT-R 6 Implicit-Each-Success-Failure-Weighted

Model structure
The model uses the implicit strategy choice structure. Rewards are given according to the each weighting scheme, as soon as part of the task is completed: after the model places a (set of) block(s) in the workspace window and either returns back to the target window to study more blocks or returns to the stop-button to stop the trial. The concept of the reward is accuracy. The magnitude of the reward is the difference between the number of blocks that has been placed in the workspace window and the number of blocks that has been forgotten (i.e., studied but not placed) after the last visit to the target window. Thus, this model incorporates both the successes of the model (how many blocks has it placed) and the failures (how many blocks has it studied but not placed).

Results
The model's performance is plotted in Figure 7.3. The model provides a bad quantitative fit to the human data with $R^2 = 0.953$, RMSE = 1.600 and SSEs of 0.66, 1.60 and 5.40 (for lockouts of respectively 0, 400 and 3200 milliseconds). The qualitative fit is also bad: the model consistently places one block independent of lockout condition.

Discussion of model performance and comparison with ACT-R 5 models
The model provides a bad fit to the human data and performs worse than most ACT-R 5 models. The fit is very similar to that of the ACT-R 5 Vanilla-Once-Weighted model. The explanation for this bad fit is the same as given for the ACT-R 6 Implicit-Once models.

7.3.4.3 ACT-R 6 Implicit- Each-Blocks-Per-Second-Weighted

Model structure
The model uses the implicit strategy choice structure. Rewards are given according to the each weighting scheme, as soon as part of the task is completed: after the model places a (set of) block(s) in the workspace window and either returns back to the target window to study more blocks or returns to the stop-button to stop the trial. The concept of the reward is time. The formula for calculating the reward magnitude is rather complex when compared to the other rewards. The motivation for using this formula and not another to calculate the impact of time on an each-weighted model is given in the section on the Explicit-Each-Blocks-Per-Second-Weighted model. The reward takes the magnitude of the number of blocks that is placed in the workspace window divided by the total time that passed between the moment the model moved the mouse to the target window to start studying and the moment it stopped placing blocks in the workspace. In a sense, this is the number of blocks that the model places per second. The range of reward magnitudes in this model is very small when compared to those of other models (between 0 and about 2.5, instead of for example between 0 and 8 in the Each-Success-Weighted models). If we would use temporal discounting on these rewards, their effect will be completely overshadowed by this mechanism. We thus decided to update the utilities of production rules without temporal discounting. Due to the small magnitude of the rewards we also had to decrease the magnitude of the utility noise to 0.1.

Results
The model's performance is plotted in Figure 7.3. The model provides a bad quantitative fit to the human data with R^2 = 0.876, RMSE = 0.806 and SSEs of 0.00, 0.33 and 1.97 (for lockouts of respectively 0, 400 and 3200 milliseconds). The qualitative fit is also bad: the model undershoots human data. It always places around two blocks per visit to the target window, independent of the lockout condition.

Discussion of model performance and comparison with ACT-R 5 models
The model provides a bad fit to the human data, and a worse fit than the best ACT-R 5 model. The current model mostly undershoots human data. It is the only variant of the implicit models that consistently places more than just one block. The explanation can be found in the disabling of the temporal difference learning mechanism. As there is no temporal discounting, the production rule that starts the rehearsing receives the same amount of utility as all the other production rules that are used during the placing of a block. The utility of the start rehearsing production rule is updated after each set of blocks that has been placed. In contrast, the utilities of the study-yth block production rules are only updated when the relevant number of blocks is placed. The model receives higher rewards if it places more blocks. Hence, the production rules that stimulate to study more blocks will receive higher rewards. The start rehearsing production rule also receives these high rewards, but its value is lowered during trials in which the model does not place a high number of blocks. As a result the model places slightly more blocks than the other implicit models.

7.4 Summary of Results
Each of the twelve models that we developed behaves in a different manner. In each case, the behavior can be explained using the characteristics of the utility learning mechanism. The implicit models do not provide good fits to the human data. The goodness of fit of the explicit models depends on the conception of the reward structure. The models that are once weighted at the end of a trial, and that get a reward that relates to aspects of time, provide the best fits. Overall, the Explicit-Once-Sum-Of-Lockout-Durations-Weighted model provides the best qualitative and quantitative fit to the human data. The fit is in some sense very similar to the fit provided by the best fitting ACT-R 5 model.

Chapter 8: General Discussion and conclusion

In the current study we have tried to model how humans shift their strategy preferences in interactive behavior, with the Blocks World task as our case study. In the Blocks World task, subjects have to copy a pattern of eight colored blocks that is depicted in a target window, by moving the blocks from a resource window to a workspace window (Gray et al., 2006, see also Chapter 3 and Figure 3.1). However, each window is covered by a gray rectangle and the window area only becomes visible when subjects move their mouse cursor into the window area. In addition, in order to view the information in the target window, subjects have to wait for a lockout time between 0 and 3.2 seconds (manipulated between subjects). The magnitude of the lockout time influences the strategies that people use. If information is easily accessible due to a small lockout time, subjects mainly use interaction with the environment to view the example pattern (i.e., they often visit the target window to study blocks). If information is harder to access due to an increased lockout time, subjects tend to study more blocks during each visit to the target window, and to place more blocks after this study phase (Gray et al., 2006). As a result, they also place more blocks after their first visit to the target window. We have modeled this task in ACT-R 6 (Anderson, 2007). Our goal was to see if models that are developed in ACT-R 6 (Chapter 7) provide better fits to human data than previous models that were developed in ACT-R 5 (Gray et al., 2005, see also Chapter 6). We will now summarize our conclusions.

8.1 Back to the Research Question

Our core research question is: "Does an ACT-R 6 model of the Blocks World task that incorporates the latest mechanism for conditioning (Anderson, 2007; Fu & Anderson, 2004, 2006) provide a better qualitative and quantitative fit to the human data than the ACT-R 5 models of that task (Gray et al., 2005)?" Our initial hypothesis was that the ACT-R 6 models provide a better fit than the ACT-R 5 models. This was motivated by the fact that ACT-R 6 has a natural way for dealing with rewards of different magnitudes. This is lacking in ACT-R 5, and has been identified as a flaw.

ACT-R 6 incorporates temporal difference learning. This algorithm tends to favor production rules that lead to rewards fast, given that all else (for example reward magnitudes) is equal (Anderson, 2007). This is very similar to the prediction of the soft constraints hypothesis. This hypothesis states that humans optimize their behavior to minimize the time that they spend on interactive routines, given that all else is equal (Gray et al., 2006). Thus, ACT-R 6 has a natural mechanism for incorporating the soft constraints hypothesis.

As the result sections of the ACT-R 6 models showed, only one of our models provided a good qualitative and quantitative fit to the human data that is about as good as the fit of the best ACT-R 5 model. Most other models had a fair or bad quantitative fit, and a bad qualitative fit. This rejects our hypothesis. However, there is a nuance. The development of a reasonable fitting ACT-R 5 model required changes in the way that expected gain is calculated in ACT-R 5. "Normal" ACT-R 5 models that do not incorporate these changes (i.e., in case of the ACT-R 5 Vanilla-Once-Weighted and Vanilla-Each-Weighted models) provided bad fits to the human data as well. In contrast, our ACT-R 6 models did not require any changes in the ACT-R architecture in order to provide a good fit (with the exception of the Each-Blocks-Per-Seconds-Weighted models). We thus tend to conclude that the ACT-R 6 models *do* provide a better qualitative and quantitative fit to the human data than the ACT-R 5 models. This confirms our hypothesis. The goodness of fit depends on the reward structure (both the magnitude of the reward and the moment when the reward is experienced) and the structure of the production rules. Our complete answer to the hypothesis therefore is as follows:

Regular ACT-R 6 models of the Blocks World task (i.e., models without changes in the architecture) can provide a better qualitative and quantitative fit to the human data than regular ACT-R 5 models of the Blocks World task. The goodness of fit of the model depends on the way in which the reward structure is represented in the model, and on the structure of the production rules. The best fits are provided by ACT-R 6 models that make single explicit strategy choices, that receive rewards once per trial at the end of the trial, and in which the magnitude of the reward is either (minus one times) the amount of time spend on the task, or (minus one times) the total time spend waiting on the lockout time. However, when compared to ACT-R 5 models in which alterations to the architecture are allowed, the ACT-R 6 models provide less good fits.

8.2 Do the ACT-R 6 models provide a better fit to the human data than the ideal performer model?

Our two best fitting ACT-R 6 models (Explicit-Once-Total-Trial-Time-Weighted and Explicit-Once-Sum-Of-Lockout-Durations-Weighted) had an R^2 of about 0.99 and RMSE's of about 0.4. These are very similar to the values of the ideal performer model (Gray et al., 2006, see also Chapter 4). This is interesting, as our ACT-R 6 models are more constrained than the ideal performer model. First, they are developed within a complete cognitive architecture. This is in contrast to the ideal performer model that is constrained to a set of side conditions (Simon, 1992), but not to a set that is as big as that of the cognitive architecture ACT-R. Second, our ACT-R 6 models have a shorter training phase than the ideal performer model: twenty-five trials instead of hundred thousand. In the ideal performer, this training phase forms a separate phase from the test phase, and different processes govern the selection of actions during the training phase and the test phase. In contrast, the "training phase" of the ACT-R 6 model is no separate phase and forms part of the normal trials during which the model's behavior is tested. The model's behavior and the selection of production rules are guided by the same mechanisms in the "training" and "test phase".

Despite these observations, we do not argue that the ACT-R 6 models provide a better fit to the human data than the ideal performer model. For one thing, the ideal performer model has shown to have a good fit on several behavioral measures of the Blocks World task (see Gray et al., 2006, for these measures). We have only compared the fit of the ACT-R 6 models on one measure (as this is also the only measure on which the ACT-R 5 models were compared). Also, the ideal performer model does not vary its actions as much as the ACT-R 6 models. Once it has learned that specific encode-x strategies have high utilities, it keeps on using them. This is similar to the human data. In contrast, our ACT-R 6 model also keeps on using low(er) encode-x strategies in situations where higher encode-x strategies might be better to use. In this sense, the ideal performer model provides a better qualitative fit to the human data than the ACT-R 6 model.

The ideal performer model is able to keep the variance in its behavior low due to the way the world is represented: as state-action pairs. For each state of the world (indicated by the number of blocks that has already been placed by the model), different encode-x actions were developed. Each of these state-action pairs has its own utility value. As a result, the ideal performer model can not only learn how useful a specific encode-x action is, but also how useful it is to use that action given a specific state of the world (for example at the beginning or the end of a trial). The ACT-R 6 models cannot distinguish between these different state-action pairs. We thought of including them, but this would require a longer learning time or training phase. This is unrealistic. ACT-R models should mimic human performance and prior knowledge, and humans learn faster than the ideal performer model.

8.3 General Conclusions for the ACT-R community

We know how our study confirms our hypothesis and what the limitations of the current models are. We will now discuss how these conclusions generalize and what their implications are for the ACT-R community and for other ACT-R models that are developed by the community.

8.3.1 How a modeler's conception of the task structure and the rewards influences a model's behavior

If there is one big thing that can be learned from our study, it is that a modeler's conception of the task and the rewards in the task can have a big influence on the model's behavior. First of all, we saw that the process by which different strategies can be chosen (in terms of the used set of production rules) influences the model's behavior. While prior knowledge, reward structures, and parameter values were kept constant between the explicit and the implicit models, both showed very dramatic differences in behavior. While some of the explicit models provided a good or fair fit to the human data, none of the implicit models provided a good or at least fair fit to the human data. In our discussion of the behavior of the implicit models we have shown how some aspects of this behavior can be explained in terms of the structure that the production rules have.

A similar conclusion holds for the reward structure of the task. We manipulated two aspects in this study: the moment when the reward is experienced and its magnitude. The moment when the reward is experienced is important for utility learning, as a production rule's utility converges towards the size of the experienced reward minus the average time between the moment the production rule fired and the moment that the reward triggered (Anderson, 2007).

In six different reward-weighting schemes we also manipulated the magnitude of the reward. Fundamentally the manipulations differed in what aspect of the task the model conceptualizes as a reward. A different conception of what a reward is, results in a different magnitude of the reward. On the one hand, a reward can be conceptualized in terms of *how good* the model performs the task itself: how many blocks does it place after a visit to the target window, and how many blocks does it study but forget? Different models have different reward functions, but in general the rewards for this class of models of the Blocks World task range between -8 (all blocks studied, none placed) and 8 (all blocks placed). A totally different conception is to express rewards in terms of *how fast* the model performs the task (or specific parts of it). Different models have different reward functions, but in general the rewards are negative: the more time the model spends on the trial (or on specific parts of it), the more negative the reward is. In this case rewards range between 0 and about -80.

As shown in the examples above, the modeler's conception of the rewards of a task has a big influence on the reward magnitude (i.e., the difference between -80 and +8). The reward magnitude has a big influence on the utility value of production rules, and this has a big influence on the behavior of a model. In the end, the modeler's conception of rewards has a big influence on the model's behavior. These different ways of conceptualizing rewards is not just specific for the Blocks World task. Many psychological tasks often distinguish between speed of performance and accuracy of performance. Moreover, it is a general issue in artificial intelligence and computer science: the way the problem is represented influences how easily automatic-learning algorithms can find the right solution (e.g., Russell & Norvig, 1995; Sutton & Barto, 1998). Cognitive modelers will think hard about the best way of representing their task and the rewards in their task. Most of the time, they will choose the right representations. However, it is good to be aware of how much impact these choices can have.

The issue that we describe here is of more importance to ACT-R 6 than to ACT-R 5. In both versions of ACT-R, a modeler can develop different models of a particular task. In both versions of ACT-R, a modeler can vary the set of production rules that is used (for example explicit and implicit models) and the moment when rewards are given (for example, in a once- or an each-weighted style). However, only in ACT-R 6 can the modeler vary the magnitude of the rewards, as the modeler is not limited to giving binary feedback.

We have identified several methods for modeling the magnitude of a reward, and have identified which methods work best for the Blocks World task. It is not desirable if all tasks require such a broad exploration of alternatives. Hence, efforts should be made to find general guidelines that can help cognitive modelers to choose the magnitude of the rewards in their task. We recommend that two alternatives be tested in a broad set of tasks: conceptualizing rewards based on accuracy of task performance and conceptualizing it as the amount of time spend on the task (or on sub aspects of the task). Our study seems to favor the latter form. Guidance might also come from explorations of neurological measures: is there a relationship between the dopamine rate in the brain and the magnitude of rewards in ACT-R models (just like there are relationships between general reinforcement learning models and dopamine rates, e.g., Schultz et al., 1997).

8.3.2 Temporal difference learning by itself does not always optimize performance time

The ACT-R 6 models provided the best fit if the reward structure represented the amount of time spend on the task. Potentially, this is because it is only in these cases that different strategy choices have a big impact on the magnitude of the reward (working longer on a task makes the reward more negative). In contrast, the reward magnitude did not differ that much between alternative strategies in models that conceptualized rewards based on accuracy (as the reward magnitudes ranged between -8 and 8). In these models the positive effect of receiving a higher reward is often even overshadowed by temporal discounting. This problem has been identified before (Anderson, 2006, 2007). If critical choices are made early on in trials, temporal difference learning cannot learn to optimize behavior. Our results seem to give a nuance to this claim. As long as the reward magnitudes of different alternatives are big enough, temporal difference learning algorithms can learn the importance of early critical production rules.

8.3.3 Determining the right learning speed

Humans seemed to learn to favor specific strategies faster than our ACT-R models. Learning effects mostly take place during the first ten trials (Gray et al., 2006). This contrasts to the ACT-R models (both in ACT-R 5 and ACT-R 6), which needed twenty-five trials to settle in on a good strategy. And even then they sometimes alternated between different strategies. There are three options for improving the speed with which the ACT-R model learns to settle in on a strategy. First of all, ACT-R's learning rate (alpha) can be set higher. However, the bigger the magnitude of the learning rate, the more impact small changes in the environment have on the utility value of production rules. This might lead to instable behavior and is unfavorable.

A second option is to start out with initial utility values that are near the eventual utility values. In some of our ACT-R models the eventual utility values were as big as -60. If these models start of with a utility value of zero, it will take several trials before the utility values reach their eventual values. The closer the initial utility values are to the eventual utility value, the less time the model spends on learning the right utility values. We do not expect that human subjects initially have different conceptions of utility values in different experimental conditions. Also, we want the model to learn the true utility values during experience with the task. Hence, the utility values of different critical production rules should initially have (almost) the same magnitude.

A third option would be to replace the overall production rule structure, and to include other types of production rules in which the models can learn what the right utilities are. We discuss several options for our own model in Appendix A8. Here we want to emphasize again that the way in which production rules are structured influences the model's behavior, and it is a general issue about which cognitive modelers have to think when developing their models.

8.4 General conclusion about Strategy Shifts in Interactive Behavior

In this thesis we have described strategy shifts as the process of learning the utility values of different strategies, and adapting the selection of strategies to use the ones with the highest utility. We have seen that this can be modeled in the cognitive architecture ACT-R, at least for the Blocks World task. The fits that the ACT-R 6 models provide to the human data are not perfect, and the goodness of fit depends on the settings for variables of the architecture. However, they do not require any alterations of the primary assumptions of the architecture. Moreover, we were able to explore some of these settings and the impact of them on the behavior of the models. Altogether it seems that the most important influence on the behavior of the models, and presumably on human behavior, is the way in which the rewards of accomplishments are represented. In cases like the Blocks World task, the speed at which the task is performed is very important, and it seems that in these cases the temporal costs of the task needs to be represented explicitly in the reward magnitude. Also, in order to optimize local aspects of a task (e.g., to optimize how many task information is memorized), it seems beneficial to update the estimate of the utility of actions (or production rules) only once, at the end of the task. This way, the impact of critical decisions on the complete task can be taken into account. Future work in cognitive science, both using cognitive models and using other means, should investigate whether these conclusions also hold for other tasks that involve interactive behavior and if they hold for other domains in which strategy shifts occur.

References

Anderson, J. R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. (1991). Is human cognition adaptive? *Behavioral and Brain Sciences, 14*(3), 471-517.

Anderson, J. R. (2006). *A new utility learning mechanism.* Paper presented at the 2006 ACT-R workshop.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review, 111*(4), 1036-1060.

Anderson, J. R., & Lebiere, C. J. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.

Anderson, J. R., & Schooler, L. J. (1991). Reflections of the environment in memory. *Psychological Science, 2*(6), 396-408.

Ballard, D. H., Hayhoe, M. M., & Pelz, J. B. (1995). Memory representations in natural tasks. *Journal of Cognitive Neuroscience, 7*(1), 66-80.

Ballard, D. H., Hayhoe, M. M., Pook, P. K., & Rao, R. P. N. (1997). Deictic codes for the embodiment of cognition. *Behavioral and Brain Sciences, 20*(04), 723-742.

Bothell, D. (2005). ACT-R 6 Official Release. *Proceedings of the 12th ACT-R Workshop*.

Bothell, D. (2008). ACT-R 6 Reference Manual: Available on the ACT-R website (http://act-r.psy.cmu.edu/actr6/).

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence Journal, 47*, 139–159.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*: Lawrence Erlbaum Associates.

Carter, C. S., Braver, T. S., Barch, D. M., Botvinick, M. M., Noll, D., & Cohen, J. D. (1998). Anterior cingulate cortex, error detection, and the online monitoring of performance. *Science, 280*(5364), 747-749.

Erev, I., & Barron, G. (2005). On adaptation, maximization, and reinforcement learning among cognitive strategies. *Psychological Review, 112*(4), 912-931.

Fu, W. T., & Anderson, J. R. (2004). Extending the computational abilities of the procedural learning mechanism in ACT-R. *Proceedings of the 26th annual meeting of the Cognitive Science Society*, 416–421.

Fu, W. T., & Anderson, J. R. (2006). From recurrent choice to skill learning: A reinforcement-learning model. *Journal of Experimental Psychology: General, 135*(2), 184-206.

Gonzalez, C., Lerch, J. F., & Lebiere, C. (2003). Instance-based learning in dynamic decision making. *Cognitive Science, 27*(4), 591-635.

Gray, W. D., & Boehm-Davis, D. A. (2000). Milliseconds matter: an introduction to microstrategies and to their use in describing and predicting interactive behavior. *Journal of Experimental Psychology: Applied, 6*(4), 322-335.

Gray, W. D., Schoelles, M. J., & Sims, C. R. (2005). Adapting to the task environment: Explorations in expected value. *Cognitive Systems Research, 6*(1), 27-40.

Gray, W. D., Sims, C. R., Fu, W. T., & Schoelles, M. J. (2006). The soft constraints hypothesis: A rational analysis approach to resource allocation for interactive behavior. *Psychological Review, 113*(3), 461-482.

Gunzelmann, G., Anderson, J. R., & Douglass, S. (2004). Orientation Tasks with Multiple Views of Space: Strategies and Performance. *Spatial Cognition and Computation, 4*(3), 207-253.

Holroyd, C. B., & Coles, M. G. H. (2002). The neural basis of human error processing: reinforcement learning, dopamine, and the error-related negativity. *Psychological Review, 109*(4), 679–709.

Larkin, J. H. (1989). Display-based problem solving. In D. Klahr & K. K. (Eds.), *Complex information processing: The impact of Herbert A. Simon* (pp. 319–341). Hillsdale, NJ: Erlbaum.

Lovett, M. C. (1998). Choice. In J. R. Anderson & C. Lebiere (Eds.), *The atomic components of thought* (pp. 255–296). Mahwah, NJ: Erlbaum.

Lovett, M. C. (2005). A Strategy-Based Interpretation of Stroop. *Cognitive Science, 29*(3), 493-524.

Miller, G. A. (1956). The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychological Review, 63*(2), 81-97.

Nason, S., & Laird, J. E. (2005). Soar-RL: integrating reinforcement learning with Soar. *Cognitive Systems Research, 6*, 51-59.

O'Doherty, J. P., & Bossaerts, P. (2008). Toward a mechanistic understanding of human decision making: contributions of functional neuroimaging. *Current Directions in Psychological Science, 17*(2), 119-123.

Rieskamp, J., & Otto, P. E. (2006). SSL: A theory of how people learn to select strategies. *Journal of Experimental Psychology: General, 135*(2), 207-236.

Roberts, M. J., & Newton, E. J. (2001). Understanding strategy selection. *International Journal of Human-Computer Studies, 54*(1), 137-154.

Russell, S. J., & Norvig, P. (1995). *Artificial intelligence: a modern approach*. Upper Saddle River, NJ: Prentice-Hall, Inc.

Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science, 275*(5306), 1593-1599.

Siegler, R. S. (1991). Strategy choice and strategy discovery. *Learning and Instruction, 1*(1), 89-102.

Simon, H. A. (1992). What is an" explanation" of behavior. *Psychological Science, 3*(1), 150-161.

Sims, C. R., & Gray, W. D. (2004). Episodic versus semantic memory: An exploration of models of memory decay in the serial attention paradigm. *Proceedings of the 6th International Conference on Cognitive Modeling*, 279-284.

Sperling, G., & Dosher, B. A. (1986). Strategy and optimization in human information processing. In *Handbook of perception and human performance: Volume 1. Sensory processes and perception*. New York, NY: John Wiley and Sons.

Sun, R. (1997). Learning, action, and consciousness: a hybrid approach towards modeling consciousness. *Neural Networks, 10*(7), 1317-1331.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say “broke”? A model of learning the past tense without feedback. *Cognition, 86*(2), 123-155.

Taatgen, N. A., & Lee, F. J. (2003). Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors, 45*(1), 61-76.

Taatgen, N. A., van Rijn, H., & Anderson, J. R. (2007). An integrated theory of prospective time interval estimation: The role of cognition, attention and learning. *Psychological Review, 114*(3), 577-598.

Tulving, E. (1972). Episodic and semantic memory. In E. Tulving & W. Donaldson (Eds.), *Organization of memory* (pp. 381-403). New York: Academic Press.

Watkins, C., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3-4), 279-292.

Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review, 9*(4), 625-636.