A Computational Unification of Cognitive Control, Emotion, and Learning

by

Robert P. Marinier III

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2008

Doctoral Committee:

Professor John E. Laird, Co-Chair
Professor Richard L. Lewis, Co-Chair
Associate Professor Satinder Singh Baveja
Associate Professor Thad A. Polk
Research Associate Professor Jonathan Gratch, University of Southern California

Dedication

For my Bun, who brings out my best emotions.

Acknowledgments

My thesis would not have been as timely or as high-quality without the technical support of several people. I would like to thank Mike Parent, for making the Soar light bulb in my head finally turn on, Karen Coulter and Scott Wallace for introducing me to kernel hacking, Doug Pearson for making SML happen, Nate Derbinsky for fixing up Soar-RL, and especially Jon Voigt, for endless programming support.

My theoretical ideas would not have been as developed without the influence of several sources. I would like to thank Jon Gratch and Stacey Marsella for providing my original inspiration, Professor Barb Fredrickson for teaching the emotions class, Professor Steve Kaplan for making me think outside the Soar box, Shelley Nason for switching topics with me and developing Soar-RL, and Allen Newell, who made me wish I was born 20 years earlier.

I wouldn't have survived grad school had it not been for the moral support of several friends. I would like to thank Andy and Sarah Nuxoll for convincing me that there's more to life than work, Masters Andy DeJesus and Jane Ophoff for teaching me the best forms of stress relief, Devvan Stokes for thoughtful conversation, and especially my wife Orsi Dézsi, who provided endless snuggles.

Finally, I would like to thank John Laird; words cannot adequately capture the positive influence he has had in all of these areas.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Research on the integration of emotion and cognition has existed for many years (Schorr, 2001). This research has made great strides in establishing that emotion and cognition are, in fact, intimately connected, and several computational models have emerged that embody these ideas (Ortony et al., 1988; Neal Reilly, 1996; Gratch & Marsella, 2004; Hudlicka, 2004). However, the integrations achieved to date are to some extent incomplete. On the one hand, the claim that cognition is a necessary antecedent to at emotion is well established, and specific cognitive mechanisms that support emotion have even been established (Smith & Kirby, 2001). However, the computational realizations of this integration have largely been pragmatic. Thus, if an emotion theory claims that some cognitive step must take place, such as determining whether a stimulus is relevant to the current goal, then a subsystem is implemented that makes it take place, with little consideration of its overall role in cognition and why it must take place. That is, the link between core cognitive functions and emotion has yet to be fully explored.

Our approach is to start with a theory of cognitive control called PEACTIDM (Newell, 1990; pronounced PEE-ACK-TEH-DIM) and show how a set of emotion theories called appraisal theories naturally fills in missing pieces in PEACTIDM, while PEACTIDM provides the computational structures needed to support appraisal theories. PEACTIDM is a set of abstract functional operations that all agents must perform in order to generate behavior (the acronym denotes these operators, described in detail below: Perceive, Encode, Attend, Comprehend, Tasking, Intend, Decode, Motor). While PEACTIDM describes the abstract operations, it does not specify the source and types of data that these operations manipulate. We claim that appraisal theories (Roseman & Smith, 2001) provide exactly the required data. Conversely, PEACTIDM provides the

functional operations missing from appraisal theories. An important consequence of this integration is that appraisals can be generated incrementally, leading to a time course of emotions. This integration is performed within the Soar cognitive architecture (Laird, 2008), but could equally apply to similar cognitive architectures such as ACT-R (Anderson, 2007). We furthermore show that the integration provides a natural basis for understanding the role of mood and feelings.

The main purpose of this thesis is to explore the feasibility and potential value of this integration. Since there are no existing integrations of this kind, a direct comparison to alternative approaches is impossible. Instead, our evaluation focuses on whether the integrated model produces behavior that is qualitatively consistent with PEACTIDM and appraisal theory. We will also address Picard's (1997) list of properties that an emotional system should have (Chapter 5).

While we rely on psychological theories to inform our approach, our emphasis is on the functional benefits (in terms of artificial intelligence) that we can derive from this theory. Possible functional benefits might include enhanced memory retrieval, physiological preparation for action, and learning. In this thesis, we explore the use of emotion as an intrinsic motivation to drive reinforcement learning. This exploration is carried out in two domains under a variety of conditions in order to tease out the impact of various parts of the model (Chapters 7-9). The integration also results in extensions to reinforcement learning, such as the source of intrinsic reward, and automatic setting of the learning and exploration rate parameters.

The remainder of this thesis is organized as follows. In Chapter 2 we provide background on cognitive and emotion theories, with a focus on PEACTIDM, Soar and Scherer's (2001) appraisal theory. In Chapter 3 we describe the unification of these in the context of a model of a simple, short task. In Chapter 4 we describe a slightly more complex model of an extended synthetic task, and in Chapter 5 we present an evaluation of that model. Chapter 6 describes related work in the context of our theory. Chapter 7 introduces the integration of emotion and reinforcement learning. Our purpose is to see if emotion can be used as the basis of a reward signal to intrinsically motivate the agent. Initial results are also presented. Based on our success there, we decided to explore

additional issues. Would the system scale to a more complex domain? How do various aspects of the system influence the performance? Specifically, what influence do various appraisals have? Chapter 8 introduces a new, more complex domain in which we explore these issues, and also describes revisions to the model based both on our earlier experience with the model and demands of the more complex domain. Chapter 9 presents a series of experiments designed to explore these issues, along with our evaluation of the results. Finally, Chapter 10 summarizes, describes future work, and concludes.

Chapter 2

Background

In this chapter, we describe PEACTIDM, a theory of cognitive control, and present background on cognitive theories, particularly Soar, in terms of PEACTIDM. We then present background on emotion theories, and make the connection between PEACTIDM and appraisal theories as complementary pieces of the cognition/emotion integration puzzle.

2.1 Cognitive systems

2.1.1 PEACTIDM: An abstract computational theory of cognitive control

PEACTIDM is a theory of cognitive control where cognition is decomposed into a set of *abstract functional operations* (Newell 1990). PEACTIDM stands for the set of eight abstract functional operations hypothesized as the building blocks of immediate behavior: Perceive, Encode, Attend, Comprehend, Tasking, Intend, Decode, and Motor. These functions are *abstract* because although many of them may often be primitive cognitive acts, they can require additional processing, whose details are not specified by Newell's theory. Furthermore, Newell did not speculate about the actual data processed by these functions.

PEACTIDM was developed from a functional analysis of immediate behavior—tasks with short timescales where interaction with the environment dominates behavior. It is consistent with human data on some tasks (Newell 1990) and prior work in GOMS-like paradigms (John, Rosenbloom & Newell 1985).

We will describe PEACTIDM via illustration with a simple immediate choice response task adapted from a task described by Newell. (As we demonstrate shortly, even a simple example like this can have an emotional component.) In the task, a subject is faced with two lights and two buttons. The lights are both within the subject's fovea. The subject's task is to focus on a neutral point between the lights and wait for a light to come on. When a light comes on, the subject must press the button corresponding to that light. The subject gets feedback that the correct button was pressed by the light turning off in response to the press. The subject's reaction time is the time it takes to turn off the light.

In PEACTIDM, *Perceive* is the reception of raw sensory inputs. In this case, the subject perceives one of the lights turning on. *Encode* is the transformation of that raw sensory information into features that can be processed by the rest of cognition. In this example, a representation is created that indicates one light has come on. *Attend* is the act of attending to a stimulus element. In this case, it is not an overt eye movement but is some type of covert attention that must select the lit light (even though the light is already foveated). *Comprehend* is the act of transforming a stimulus into a task-specific representation (if necessary) and assimilating it into the agent's current understanding of the situation, such as classification or identification. In our example, the subject verifies that one of the two lights has come on (that is, his attention was not drawn by some other stimulus). *Tasking* is the act of setting the task (i.e., the goal) in the internal cognitive state. In our example, Tasking takes place in an earlier cycle before the task begins—the subject is already poised, looking at the lights with a finger ready to press a button and knows which button to press for which light. It is via Tasking that Comprehend knows what to expect and Intend knows what operation to choose based on the input. Given the task and the comprehension of the stimulus, *Intend* initiates a response, in this case, pressing a button. *Decode* translates the response from Intend into a series of motor actions. *Motor* executes the action; in our example, the pressing of the button. In general, Comprehend, Intend, and Tasking may require an arbitrary amount of processing to perform their functions, although in this task very little processing is required to support those functions.

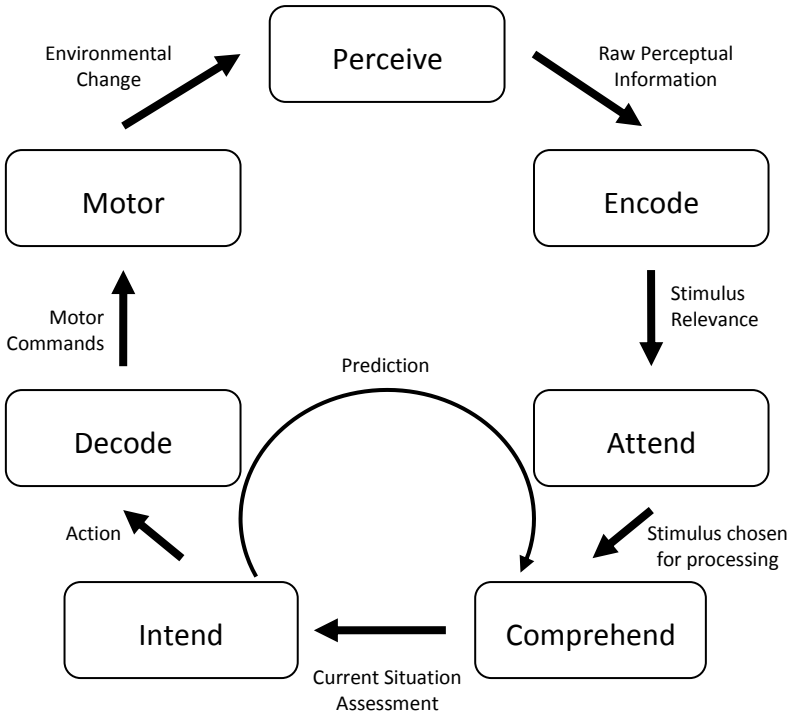


Figure 2.1: Basic PEACTIDM cycle.
The agent repeats this cycle forever. The output from a step primarily feeds into the next step, but the output of Intend also feeds into the next cycle's Comprehend. Tasking (not shown) competes with Attend. Tasking modifies the current goal, which also serves as an input to the Encode and Comprehend cycles.

Newell argued that the ordering of PEACTIDM functions is determined largely by the data dependencies between the functions (see Figure 2.1). Perceive must occur before Encode, which must occur before Comprehend, which must occur before Intend, which must occur before Decode, which must occur before Motor. In some simple cases, the presence of a stimulus is all that is required for the task, and thus the Encoding step may be skipped. Tasking is the most flexible. In the implementation presented here, Tasking competes with Attend. That is, the agent can either Attend (and thus complete the cycle as shown in Figure 2.1), or it can Task (in which case it immediately precedes to Perceive to restart the cycle). An alternative approach has it compete with Intend (see section 8.3.2).

2.1.2 Approaches to cognitive modeling

Although PEACTIDM describes a set of abstract operations, it does not describe which mechanisms realize these operations and different approaches to cognitive

modeling suggest different mechanisms. The *cognitive architecture* approach we pursue here decomposes cognition into more primitive computational components that are the building blocks for functional capabilities. The interactions among these components give rise to temporal dynamics within the system. A typical cognitive architecture consists of memories (both long-term and short-term) with different performance characteristics. For example, memories can differ what type of knowledge is stored/learned, how knowledge is represented in the memory, how it is learned, and how it is retrieved. There can also be processing components that combine knowledge, such as to select between alternative interpretations or intentions. Most cognitive architectures also have perceptual and motor systems. Thus, a cognitive architecture provides task-independent structure and subsystems that is shared across all tasks, while using task-dependent knowledge to specialize behavior for a given task. Cognitive architectures are essentially computational systems for acquiring, encoding and using knowledge.

A cognitive architecture implements PEACTIDM by implementing the abstract operations via a combination of its subsystems and knowledge that directs the interactions of those subsystems. We have chosen Soar, to realize PEACTIDM, although it should be possible to implement it in other architectures such as ACT-R (Anderson, 2007), EPIC (Kieras & Meyer, 1997), or Clarion (Sun, 2006).

In section 2.1.4.1 we will sketch how PEACTIDM might be realized in other architectures in the context of the immediate choice response task.

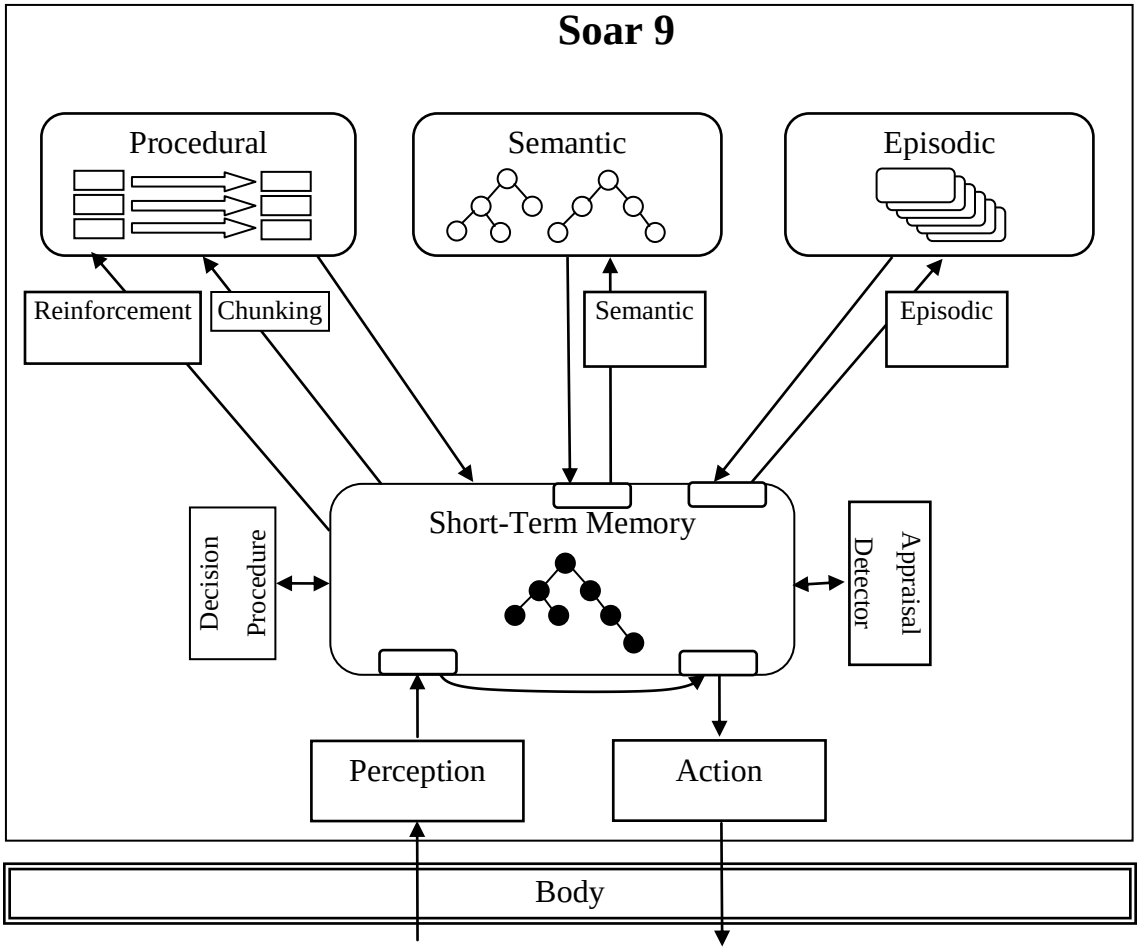


Figure 2.2: The Structure of Soar.

2.1.3 Soar

Soar is a cognitive architecture that has been used both for cognitive modeling and for developing real-world application of knowledge-rich intelligent systems. Figure 2.2 is an abstract block diagram of Soar, which shows the major memories (rounded edges) and processing modules (square edges). In the bottom middle is Soar's short-term memory (often called its working memory). The short-term memory holds the agent's assessment of the current situation, derived from perception (lower middle) and via retrieval of knowledge from its long-term memories. It has three long-term memories: procedural (production rules), semantic, and episodic, as well as associative learning mechanisms. In this work, the semantic and episodic memories are not used, but we will

return to them in our discussion of future work. The appraisal detector will be discussed in section 3.4.

Soar avoids the use of syntax-based conflict resolution mechanisms of traditional rule-based systems by firing all matched rules in parallel and focusing deliberation on the selection and application of *operators*. Proposed operators are explicitly represented in working memory, and deliberation is possible through rules that evaluate and compare the proposed operators. Soar follows a decision cycle (Figure 2.3) which begins with an Input phase in which the agent gets input from the environment. This is followed by the Propose phase in which rules fire to elaborate knowledge onto the state, and propose and compare operators. Next, based on the structures created by those rules, Soar selects an operator in the Decide phase and creates a structure in short-term memory representing the chosen operator. This choice may be determined by the comparison knowledge, or it may be random. Once an operator has been selected, rules with knowledge about how to Apply that operator can fire. Some of these rules may generate output commands. Finally, Output is processed (e.g., the world is updated in response to an action).

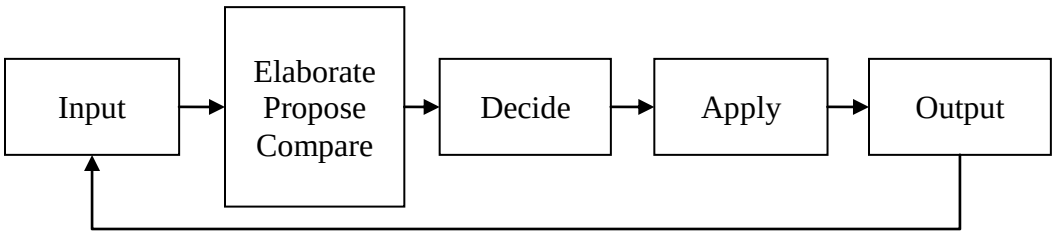


Figure 2.3: The Soar decision cycle.

Sometimes, there may not be sufficient knowledge to apply an operator. This is called an impasse. When that happens, Soar creates a substate structure in Short-Term Memory. This structure allows the Soar decision cycle to continue by allowing rules to match in the substate leading to the selection and application of suboperators that can help the agent resolve the impasse. For example, it may be that the operator is abstract, in that it requires many discrete steps (i.e., suboperators) to implement. Thus, the substate supplies a context in which the suboperators that execute these steps can be

selected. When the final step completes the application, the impasse is resolved and Soar can select a new operator in the top state. Soar can also learn one or more new rules that summarize the processing done in the substate via a process called *chunking* (Laird, Rosenbloom & Newell, 1986), allowing the impasse to be skipped in the future, which results in faster processing.

2.1.4 Implementing PEACTIDM in Soar

In this section, we walk through the simple immediate choice response task presented earlier (section 2.1.1) and describe how it is possible to map PEACTIDM onto Soar. This implementation closely following Newell's (1990) description.

Recall the task situation: the agent is faced with two lights and two buttons; the task is to press the button corresponding to the light that comes on. Before the task even begins, the agent does *Tasking*, which creates a structure in short-term memory describing the goal, which includes a prediction that a light is going to come on. *Perceive* is the reception of raw sensory inputs; in Soar this means that a structure describing which light comes on is created in short-term memory. This structure causes *Encoding* rules in procedural memory to match and generate domain-independent augmentations are added (e.g., the light coming on means the agent can make progress in the task). Rules in Soar fire in parallel, so if there were multiple stimuli, an encoded structure would be generated for each at the same time. *Attend* is implemented as an operator; this is natural since PEACTIDM only allows one stimulus to be Attended to at a time, and Soar only allows one operator to be selected at a time. Thus, there will be one proposed Attend operator for each stimulus; which one is selected is influenced by the Encoded information. In this task, only one Attend operator is proposed (since there is only one stimulus). *Comprehend* is implemented as a set of operators; exactly how many are required depends on the complexity of the task and situation. In this task, there is only one Comprehend operator, which verifies that the stimulus is what was expected (as determined by Tasking earlier). *Intend* is implemented as an abstract operator. Thus, an impasse is created and a set of operators that work together resolve the impasse by selecting a response (in this task, to push the button) and creating a prediction of the outcome of that action (in this task, that the light will turn off). In Soar, *Decode* is

merely sending the selected action to the output system, and *Motor* is handled by the simulation of the environment.

Soar naturally supports the various kinds of processing required by PEACTIDM: fast Encoding via parallel rule firing, single stimulus selection via operators, and arbitrary processing via operator chaining and impasses. Soar can even learn to compress the arbitrary processing via its chunking mechanism, increasing reactivity (see section 3.8 for an example).

2.1.4.1 Implementing PEACTIDM in Other Cognitive Architectures

As we can see, PEACTIDM fits naturally with Soar. Earlier, however, we claimed that there was no inherent connection between PEACTIDM and Soar, and that PEACTIDM could also be implemented in other cognitive architectures such as ACT-R (Anderson 2007), EPIC (Kieras & Meyer, 1997), or Clarion (Sun, 2006).

We will give a brief sketch describing how PEACTIDM might be implemented in ACT-R, highlighting the key capabilities necessary to achieve this. ACT-R is composed of several modules, each of which performs a distinct function. The declarative memory module stores a copy of everything the agent thinks about. These memories can be retrieved later via an explicit retrieval, which returns the best partially matching memory. The goal module contains the agent's current goal structure. The imaginal module is like a scratch pad the agent can use to store any structure. The perceptual and motor modules connect the agent to the environment. Each of these modules has an associated buffer that exposes its information to the system (somewhat analogous to Soar's Short-Term Memory). For example, the goal module's buffer simply contains the goal structure, whereas the declarative memory module's buffer can contain a cue, which triggers a retrieval that overwrites the cue. A rule module matches rules based on the contents of the buffers and selects a single rule to fire next (if multiple rules match, a conflict resolution mechanism chooses one).

Perceive may be naturally implemented via ACT-R's perceptual module. In general, the necessary ability is to bring external information into the architecture. Encode needs the ability to transform or augment this information so that it is in a form

that the rest of the system can easily use. We utilized Soar's ability to fire multiple rules in parallel in order to generate multiple encodings. ACT-R can only fire one rule at a time, making it difficult to map Encoding onto specific cognitive functions in ACT-R. Given the level of output of the perceptual systems in ACT-R (e.g., words), a more reasonable mapping would be to have Encoding handled by the perceptual module, so that when external information is available to cognition, is it already Encoded. Attend requires the ability to choose a single Encoded structure for further processing. In ACT-R, since only one rule can fire at a time, a separate rule can be associated with each Encoded structure that creates appropriate structures in one of the available buffers (either goal or imaginal). The competition between possible Attend rules in ACT-R is analogous to the competition between Attend operators in Soar. Comprehend can also be implemented as a series of rule firings, some of which may induce retrievals from declarative memory. For example, the verify operator in Soar may be implemented as an attempt to retrieve a prediction structure that matches the current situation. This actually allows for at least three levels of match: if the retrieval fails, then the prediction was not even close, whereas if the retrieval succeeds, then the prediction was at least close. An additional step via a rule can determine if the prediction matches exactly or not. A possible issue in this would be if an old prediction was retrieved, and not the most recent one. If temporal information is stored with a prediction, then a rule may be able to determine if this has happened and classify accordingly (e.g., no match at all). Tasking can be implemented as rules or retrievals that ultimately create and modify structures in the goal buffer. Intend may be directly implemented as a rule in ACT-R, or it may be a subtask (similar to how Soar uses a substate). Decode and Motor may be handled by ACT-R's motor module.

As we can see, some aspects of ACT-R potentially make PEACTIDM more difficult to implement (e.g., the Encode function), but other aspects may be more realistic, or suggest alternative implementations in Soar. For example, ACT-R's perceptual and motor modules may improve Perceive and Motor; Soar has been extended with similar capabilities before (Chong, 1997), which would improve the realism of perception and action modeled in this research. Using ACT-R's partially matching

declarative memory for Comprehend also suggests a similar route for Soar via its semantic memory.

2.1.5 Architectural Requirements of PEACTIDM

Thus, our usage of PEACTIDM has revealed possible strengths and weaknesses in both Soar and ACT-R. For example, in Soar, we are required to abstract away from the details of Perception and Motor, whereas in ACT-R, reliance on the perceptual module for Encoding means that ACT-R cannot learn Encoded structures[1]. In general, the requirements of PEACTIDM may expose the strengths and weaknesses of other existing architectures as well. Table 2.1 reviews the architectural requirements for each PEACTIDM function and how Soar and ACT-R support them.

Function	Architectural Requirements	Soar	ACT-R
Perceive	Support perception	Abstracted into environment interface	Part of perception module
Encode	Generate multiple structures in parallel	Parallel rule firings	Part of perception module
Attend	Single out one stimulus for additional processing	Operator selection	Rule firing
Comprehend	Support arbitrary processing ranging from short (e.g., verify) to long (e.g., understanding complex relationships)	Sets of operators, possibly in impasses	Sets of rules; declarative memory
Tasking	Support making persistent changes in memory	Operator application rules	Goal buffer
Intend	Support arbitrary processing ranging from short (e.g., push button) to long (e.g., complex, temporally extended action); support creating predictions	Sets of operators, impasses	Sets of rules
Decode	Expand compact command descriptions into complex motor sequences; possibly support parallel processing of multiple commands	Parallel rule firings; abstracted into environment interface	Part of motor module
Motor	Support motor execution	Abstracted into environment interface	Part of motor module

Table 2.1: Summary of the architectural requirements for PEACTIDM.

[1] To be clear, we make no attempt to learn Encoded structures in Soar, either, but as Encoding is integrated with Soar's central mechanisms, this exploration is possible without resorting to architectural modifications.

An additional requirement not listed above is the ability to learn. In principle, learning can take place in all functions except perhaps Perceive and Motor. Soar and ACT-R have comparable learning mechanisms that cover at least many of the kinds of learning possible in PEACTIDM. We discuss learning in our model in section 3.8 and chapters 7-9.

Finally, we can discuss what flexibility an architecture gives up by committing to PEACTIDM. At some level, the answer may be nothing—the functions can be viewed as a recasting of existing functionality. But it does suggest that certain kinds of processing would be inappropriate. For example, it suggests that the architecture should not support executing physical actions directly from, say, Comprehend. Perhaps more generally, it suggests that different structures should be involved in different steps (e.g., a Comprehend structure is not also a Motor command). But given that these functions are abstract, multiple possible mappings to any architecture are likely to work, and without more detailed specifications, it would be difficult to definitively exclude particular mappings.

2.1.6 What PEACTIDM and cognitive architectures provide

PEACTIDM provides constraints on the structure of processing that are more abstract than cognitive architectures like Soar or ACT-R. While Soar and ACT-R specify processing units, storage systems, data representations, and the timing of various mechanisms, they are only building blocks and by themselves do not specify how behavior is organized to produce immediate behavior. PEACTIDM specifies the abstract functions and control that these components must perform in order to produce intelligent immediate behavior.

Some of the key constraints that arise from the combination of PEACTIDM and cognitive architectures are:

- The set of computational primitives that behavior must arise from (Cognitive architecture)
- The temporal dynamics of cognitive processing and behavior (Cognitive architecture & PEACTIDM)

- The existence of core knowledge and structures that must be reused on all tasks (Cognitive architecture & PEACTIDM)

The principle theoretical gain in positing and appealing to a level of analysis at the abstract functional operator level is that it identifies common computational functions across a wide range of tasks. It thus provides a level of description at which a range of regularities may be expressed concerning the nature of these functions. We now exploit this level of description by showing how the inputs and outputs that these operators require implies that they must in fact constitute an affective system of a kind assumed in appraisal theories of emotion.

2.2 Emotion modeling

2.2.1 What can emotion provide?

PEACTIDM and cognitive architectures describe processes and constraints on representation and the timescale of those processes, but they do not describe the specific knowledge structures that are actually used to produce behavior—it is up to the modeler to describe those, and the space of possibilities is large. Consider PEACTIDM: What structures does Encode generate? Given multiple stimuli, what information does Attend use to choose which to focus on? What information does Comprehend generate? What information does Intend use to generate a response? We propose that much of the information required by PEACTIDM is generated by the same processes that generate emotion, and that these processes are, in fact, the PEACTIDM operations themselves. The abstract functions of PEACTIDM need information about relevance, goals, expectations, and so on, and compute them to carry out their functions. The results of these computations, then, cause an emotional response.

2.2.2 Introduction to appraisal theories

The hypothesis that there is a relationship between the way someone interprets a situation (along certain dimensions, such as Discrepancy, Outcome Probability, and Causal Agency) and the resulting emotional response is a defining characteristic of *appraisal theories*. Appraisal theories argue that emotions result from the evaluation of the relationship between goals and situations along specific dimensions (see Roseman &

Smith 2001 for an overview). For purpose of understanding the functional role of emotion in cognitive architectures, appraisal theories are appealing because they are naturally described at the cognitive level, as opposed to the neurological or sociological levels. Smith & Lazarus (1990) argued that, in general, emotions allow for a decoupling between stimulus and response, which is required to allow organisms to adapt to a broader range of situations. This decoupling, then, meant that more complex cognition was required to fill in the gap. In other words, complex cognition goes hand-in-hand with complex emotion. Thus, it has been claimed that one of the primary functions of more complex cognition is to support appraisal generation (Smith & Lazarus, 1990).

Appraisal theories fit naturally into our immediate choice response task. When the subject presses the button, he Encodes the state of the light and Attends to it. In the Comprehend stage, he verifies that the light's state matches his prediction. Suppose that after the first several trials, the experimenter disables the buttons so that the light stays turned on even when the correct button is pressed. When the subject Intends pressing the button, he still creates the same prediction—that the light will turn off. When the subject presses the button, though, the light does not turn off. Thus, when the subject gets to the Comprehend step, he will detect a mismatch between the actual state and the expected state.

This mismatch is called Discrepancy from Expectation, and the subject generates a structure to represent it. If the subject has high confidence in an unmet prediction, it might react differently from when the subject has low confidence in an unmet prediction. Thus, when the subject generates the prediction, an Outcome Probably is also generated. In this case, since the subject had no reason to suspect that the light would not turn off when the correct button was pushed, the Outcome Probability was very high.

Since the Discrepancy from Expectation in this case conflicts with the Outcome Probability, we expect the subject would experience surprise. The subject may not even believe what just occurred, and try to press the button again, going through the same steps. However, the second time through, the Outcome Probability is probably lower, and certainly after a few tries, the subject will realize that the button is not functioning. Emotionally, the subject's reaction may vary based on many factors, such as who he

thinks is at fault (which we call the Causal Agent). If he thinks he broke the button, he might feel shame. If he thinks he is being thwarted by the researcher, he might feel anger (especially if there was supposed to be some reward based on his performance).

Appraisal theories are complementary to the general cognitive model we described in that they provide a description of the data being processed by cognition. Integration with cognitive architecture can provide the mechanisms and processes that lead to appraisals and which utilize the results of appraisal (e.g., emotions, moods and feelings; see sections 2.2.3 and 4.2).

2.2.3 Scherer's appraisal theory

Just as we have chosen to implement our model in a specific cognitive architecture, Soar, we have also chosen a specific appraisal theory to work with: that proposed by Scherer (2001). We do not have a strong theoretical commitment to Scherer model, and we have chosen it largely because of the extensiveness of the theory. Most appraisal theories have six to eight appraisal dimensions, while Scherer's theory has sixteen appraisal dimensions. Thus, in the long run, if we can model Scherer's theory, there is less chance of us missing some important dimension than if we started with a simpler, possibly less complete theory.

Scherer sixteen appraisal dimensions are shown in Table 2.2. These dimensions are divided into four groups: relevance, implication, coping potential and normative significance. The columns are modal emotions—typical labels assigned to regions of appraisal space close to the sets of values shown.

Table 2.2: A mapping from appraisal dimensions to modal emotions with dimensions grouped by function (adapted from Scherer 2001).
Those dimensions in italics are not implemented in our current model. Open cells mean all values allowed. Abbreviations: Unfamiliar = Unfamiliarity, Unpredict = Unpredictable, Conducive = Conduciveness, med=medium, intent = intentional, neg = negligence, Enjoy=Enjoyment, Disp=Displeasure, Cont=Contempt, Anx=Anxiety, Wor=Worry, ang=anger, Bore=Boredom, Indiff = Indifference.

	Enjoy/ Happiness	Elation/ Joy	Disp/ Disgust	Cont/ Scorn	Sadness/ Dejection	Despair	Anx/ Wor
Relevance							
Novelty							
Suddenness	low	high/med			low	high	low
Unfamiliar			high		high	very high	
Unpredict	medium	high	high			high	
Intrinsic Pleasantness	high		very low				
Goal Relevance	medium	high	low	low	high	high	med
Implication							
Cause: Agent				other		other/ nature	other/ nature
Cause: Motive	intent	chance/ intent		intent	chance/ neg	chance/ neg	
Outcome Probability	very high	very high	very high	high	very high	very high	med
Discrepancy from Expectation	low					high	
Conducive	high	very high			low	low	low
Urgency	very low	low	med	low	low	high	med
Coping potential							
Control				high	very low	very low	
Power				low	very low	very low	low
Adjustment	high	medium		high	medium	very low	med
Normative significance							
Internal Standards Compatibility				very low			
External Standards Compatibility				very low			

	Fear	Irritation/ Cold ang	Rage/ Hot ang	Bore/ Indiff	Shame	Guilt	Pride
Relevance							
Novelty							
Suddenness	high	low	high	very low	low		
Unfamiliar	high		high	low			
Unpredict	high	medium	high	very low			
Intrinsic Pleasantness	low						
Goal Relevance	high	medium	high	low	high	high	high
Implication							
Cause: Agent	other/ natural		other		self	self	self
Cause: Motive		intent/ neg	intent		intent/ neg	intent	intent
Outcome Probability	high	very high	very high	very high	very high	very high	very high
Discrepancy from Expectation	high		high	low			
Conducive	low	low	low			high	high
Urgency	very high	medium	high	low	high	med	low
Coping potential							
Control		high	high	med			
Power	very low	medium	high	med			
Adjustment	low	high	high	high	medium	med	high
Normative significance							
Internal Standards Compatibility					very low	very low	very high
External Standards Compatibility		low	low			very low	high

The Relevance dimensions relate to what the agent should be paying attention to. Suddenness is perceptual in nature; it reflects the extent to which a stimulus is intense or has a rapid onset. Unfamiliarity and Unpredictability characterize the stimulus in the context of the agent's experience: Unfamiliarity is the extent to which this stimulus is different than other things the agent has seen before, and Unpredictability is the extent to

which the stimulus could not have been predicted. Intrinsic Pleasantness is how pleasant the stimulus is, independent of the current goal. Goal Relevance is how important the stimulus is with respect to the current goal (in a good or bad way).

The implication dimensions describe the agent's understanding of the situation. Causal Agent and Motive describe who caused the situation and why. Outcome Probability and Discrepancy from Expectation are related to explicit predictions about the situation; that is, how likely the prediction was thought to be, and to what extent it was accurate. Conduciveness is how good or bad the situation is with respect to the current goal. Urgency describes the extent to which immediate action is required.

The Coping Potential dimensions describe the agent's ability to deal with the situation. Control is the extent to which anyone can change the situation, whereas Power is the extent to which the agent can change the situation. Adjustment is the agent's ability to deal with the situation if it doesn't change.

Finally, the Normative Significance dimensions describe social aspects of the situation. External Standards Compatibility is the extent to which the situation is in line with cultural and social norms, whereas Internal Standards Compatibility is the extent to which the situation is in line with personal norms (e.g., personal morals and values).

Scherer's model differs from many appraisal theories in that it assumes a continuous space of emotion as opposed to categorical emotions. Like all appraisal theories, Scherer provides a mapping from appraisal values to emotion labels, but he describes these labels as *modal* emotions—that is, common parts of the emotion space. Given that the majority of existing computational models are categorical (Gratch & Marsella, 2004; Neal Reilly, 1996; Hudlicka, 2004), exploring a continuous model may help clarify the benefits and challenges of such a model. Furthermore, while our theory is continuous, it would be trivial to add categorical labels to regions if desired. Indeed, we introduce a labeling function later that does this (although we use it purely for analysis; see sections 3.4 and 5.1.1).

Another way in which Scherer's theory differs from most is that he proposes that appraisals are not generated simultaneously. Rather, he claims that appraisals are

generated in the order of the groupings given above for efficiency reasons. For example, there is no sense in wasting resources on computing the implications of a stimulus if the stimulus is irrelevant. We will return to this point after we have described our specific model.

Scherer also proposes a process model describing how, at an abstract level, the appraisals are generated and how they influence other cognitive and physiological systems, but it does not provide details of all the data needed to compute the appraisals, nor the details of those computations. Our computational model describes the details. Since the computational details include new constraints on how the model as a whole works, our model differs in some ways from Scherer's theory. This arises in part because of the need to develop a computational model of generation, and also because of the more limited scope of our model. Scherer's theory pays some attention to the physiological and neurological aspects of emotion, but like most appraisal theories, does not include detailed mappings from the theory to specific behavioral data or brain structures. Our model does not include a physiological or neurological model, and does not yet attempt to mode indirect influences on cognition or action tendencies. While these are excellent candidates for future work, our primary focus here is on the generation of appraisals in the context of PEACTIDM, and how appraisals influence behavior; thus, a symbolic cognitive approach is most appropriate.

Chapter 3

Theory and Implementation of Integration

The main theoretical proposal is that cognitive and behavioral control, as characterized by PEACTIDM, requires appraisal information, and that this appraisal information is computed directly by the PEACTIDM operations themselves. The generation of appraisals, and their accompanying emotional responses, then, is a byproduct of the system's normal operation. In this section, we provide the details of the integration of PEACTIDM and appraisal theory, building on Scherer's (2001) theory as described above (Table 2.2), though it should be possible to apply other comprehensive appraisal theories in a similar way.

In this chapter and Chapter 4, we describe aspects of our theory using examples. In this section, we continue to use the simple choice response task described earlier to give a detailed account of how this integration is realized. Thus, we address how appraisals and emotion are generated and over what time course, how they are represented, how emotion intensity is calculated, and the influence of expectations. Chapter 4 demonstrates how the model works in a more complex, extended task that we will use to demonstrate additional appraisals and introduce mood, feeling and their behavioral influences.

The simple choice response task follows the steps outlined in Table 3.1. This version has been slightly extended past our previous description to show what happens immediately following the button push. The times for Perceive, Decode and Motor are taken from Newell (1990). Steps implemented as operators (as described in section 2.1.4) take 50 milliseconds (the assumed timing of the Soar decision cycle).

Time (ms)	PEACTIDM Processing	Notes
0		Light on
40	Perceive	
90	Encode+Attend	
140	Comprehend	Verifies prediction
190	Intend	(to push button)
240		Impasse
290		Create prediction
340		Push button
420	Decode+Motor	Light off
460	Perceive	
510	Encode+Attend	
560	Comprehend	Verifies prediction
610	Tasking	(to mark task complete)

Table 3.1: PEACTIDM steps to the simple choice reaction task.

To summarize this extended version of the task, the light comes on, and the agent Perceives, Encodes and Attends to the light, and Comprehend verifies that this is what is expected. It then Intends to push the corresponding button. Intend is implemented as an abstract operator whose impasse is resolved by a set of operators in Soar that work together to both generate the push button command and create a prediction (that the light will go off). After this command is decoded and physically executed, the light turns off. This change is Perceived, Encoded and Attended, followed by Comprehension. Finally, the agent marks the task complete.

In the process of performing these PEACTIDM steps for this task, appraisal values are generated, which produce an emotional reaction. In this task, only a subset of the appraisals are relevant, namely Suddenness, Goal Relevance, Conduciveness, Outcome Probability, and Discrepancy from Expectation. Figure 3.1 shows the relationship between PEACTIDM and appraisal generation and which appraisal information influences which steps in the PEACTIDM process.

Perceive and Encode generate relevance appraisals, which are used by Attend. Comprehend generates assessment appraisals which are used by Intend. Intend generates the Outcome Probability appraisal, which is used by Comprehend in the next cycle. Tasking (not shown) is influenced by the current emotional state (not shown), which is determined by the appraisals. Critically, our claim is that the PEACTIDM steps require

this appraisal information in order to perform their functions, and thus it must be generated by earlier steps.

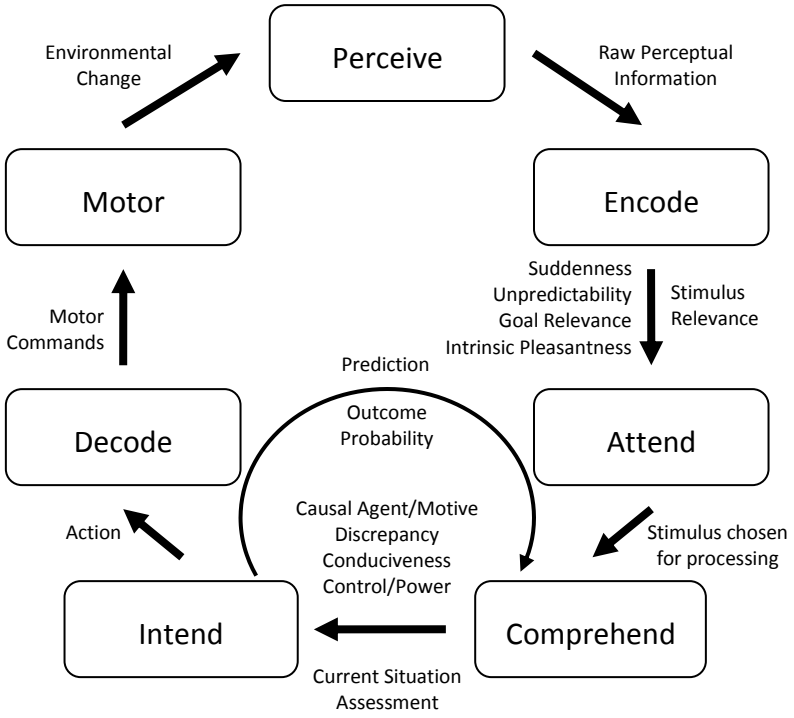


Figure 3.1: The PEACTIDM cycle with corresponding appraisals. Suddenness and Unpredictability are actually generated by Perceive, but like other pre-Attend appraisals, are not active until Attend.

3.1 Appraisal Values

The appraisals differ not only in how they are generated, but also in the types and ranges of values they can have with some appraisal values being numeric, while others are categorical. Table 3.2 shows the ranges of values we have adopted for the appraisals in our system.

Suddenness [0,1]	Unpredictability [0,1]
Goal Relevance [0,1]	Discrepancy from Expectation [0,1]
Intrinsic Pleasantness [-1,1]	Outcome Probability [0,1]
Conduciveness [-1,1]	Causal Agent [self, other, nature]
Control [-1,1]	Causal Motive [intentional, negligence, chance]
Power [-1,1]	

Table 3.2: Appraisal dimensions with ranges.

For the numeric dimensions, most existing computational models use the range [0, 1] (e.g. Gratch & Marsella, 2004). The implication is that the 0 end of the range is less intense than the 1 end of the range. For some dimensions, this is true: a stimulus with Suddenness 1 would be considered more sudden that a stimulus with Suddenness 0. For other dimensions, though, being at the "low" end could be just as intense as being at the "high" end. For example, if I pass an exam, I will appraise this as high Conduciveness and have a strong positive feeling. However, if I fail the exam, I will appraise this as very low Conduciveness, (i.e. highly unconducive) and will experience a strong negative feeling. Thus, for these dimensions we use the range [-1, 1]—that is, values near zero (e.g. not very conducive or very unconducive) would have a low impact on feeling, but values near the extremes (e.g. very conducive or very unconducive) would have high impact on feeling.

3.2 Computing the Active Appraisal Frame

In the following sections, we trace the generation of appraisals in our example. To make the calculations easier to follow, we will use extreme values, such as 1.0, for the appraisals, even though less extreme values would be more realistic.

In our example, before the task began (perhaps when waiting for the light to come on), the agent engaged in Tasking which did two things: it created a structure representing the task and a prediction structure that a light will come on. This prediction structure has an associated Outcome Probability appraisal value, which we assume is the extreme value, 1.0. When the light comes on, Perceive generates a value for the Suddenness appraisal, with value 1.0. Then, during Encoding, a structure is created with the following information: which light came on (which is domain-dependent), and whether this stimulus is on the path to completing the task. The fact that a light came on leads to a Goal Relevance appraisal value of 1.0.

The appraisals are stored in an *appraisal frame*, which is the set of appraisals that describe the current situation that the agent is thinking about it (Gratch & Marsella 2004). Before an agent Attends to a stimulus, there may be several appraisal frames that have

been started—one for each stimulus the agent perceives. We call these the pre-attentive appraisal frames.

Attend then uses the available appraisal frames to select the stimulus to Attend to. For example, the stimulus that is most Sudden may be preferred. (See the connection between Encode and Attend in Figure 3.1). When a stimulus is Attended, a flag marks the associated appraisal frame as the *active frame*. Once a frame becomes active, several other appraisals can occur. This is in line with our hypothesis that Comprehension follows Attend, and that Comprehension generates the data necessary for further processing (e.g., Intending an action; see the connection between Attend and Comprehend and Tasking in Figure 3.1). Specifically, the calculation that the stimulus is on the path to the goal leads to a Conduciveness value of 1.0.

What distinguishes our use of appraisal frames from Gratch & Marsella (2004) is that we use a single active frame to limit which appraisals are generated, whereas they have multiple complete frames; computationally, this makes our approach more efficient. Additionally, while Gratch & Marsella also use the appraisal frame to inform an attention-like process, our approach implies limits on what information can actually influence that process.

3.3 Sequences and Time Courses of Appraisals

Now that we have described how appraisals are generated, we will discuss the implications of that process on the sequencing and time course of appraisals. Scherer (2001) proposes that the appraisals are generated sequentially because the outcomes of some appraisals obviate the need for others. For example, if none of the relevance appraisals indicates that a stimulus is interesting, then there is no need to continue processing the stimulus. Our model also imposes sequential constraints (see Figure 3.1), but for two reasons, one of which is related to Scherer's. Attend will not choose a stimulus unless one of the relevance dimensions indicates that it is interesting, much like Scherer's theory describes. However, additional ordering constraints arise from the flow of data in the model. For example, since Discrepancy from Expectation arises from the Comprehension function, it occurs after the Conduciveness appraisal (which is activated

upon Attending). Similarly, the Outcome Probability appraisal is generated in the Intend step, which comes after Comprehension. Thus, while Scherer's argument for sequential appraisal generation centers on efficiency and the wastefulness of generating irrelevant appraisals, our data-driven model extends that to also impose an ordering based on data-driven constraints: the appraisals cannot be generated earlier (regardless of the efficiency). The idea of appraisals being data-driven has been mentioned elsewhere (see Roseman & Smith 2001 p.12-13 for a brief overview of this point), but the idea has been used to argue that appraisal ordering is not fixed at all. Data-driven processing combined with PEACTIDM implies at least a partial ordering.

A corollary to this is that some appraisals take longer to generate than others. In the implementation, all appraisals are generated by rules that test features of the agent's internal state, and thus fire as soon as possible. However, the amount of time it takes to generate the required features varies. As just stated, the Discrepancy from Expectation appraisal rule cannot fire until the required information has been generated by Comprehend (which in turn requires that the Attend operator has been executed). A more complex model might require an arbitrary amount of processing to generate the information necessary so that a Causal Agent appraisal rule can fire. In general, the amount of processing required by the Comprehend, Intend, and Tasking steps to enable the generation of various appraisals may be arbitrary, which is consistent with the inference vs. appraisal distinction made by Marsella & Gratch (in press). Thus, the model not only implies partially ordered sequences of appraisals, but it also implies varying time courses for the generation of those appraisals.

3.4 Determining the Current Emotion

Appraisal theories claim that appraisals are precursors to emotion (see Table 2.2). Given the theory we have described so far, it may seem that appraisal alone is sufficient. However, as we will see in Chapter 4, emotion has functional value beyond appraisal, in that it represents situation knowledge in a task-independent form that can be used to influence control and hence behavior. Here we will simply describe the emotion mechanism.

A mechanism called the Appraisal Detector (Smith & Kirby 2001) processes the active frame to determine the current emotion. It is via this mechanism that the active frame affects the rest of the system. Emotion theories disagree as to how many emotions a human can have at once. Our current model supports one active appraisal frame at a time, and thus only one emotion (not to be confused with mood or feeling, which are separate; these will be discussed in Chapter 4). The pre-attentive appraisals generated for the other stimuli do not influence the current emotion in our model.

In many systems (Ortony et al, 1988), the emotion is reported as a label (such as anger, sadness, joy, …) with an intensity. These *categorical* theories of emotion assume that there are a small, fixed number of possible feelings that vary only in intensity. In our model, like in Scherer's (2001) theory that inspires it, each unique appraisal frame corresponds to a unique experience. Categorical, linguistic labels can be generated by segmenting the space of appraisal frames, and we do this for our own analytical purposes. However, the current model does not use these labels, and even if it did, at best such labels would be a model of how an individual in a particular culture might label the emotions. For example, in the current problem, since Conduciveness and Goal Relevance are positive, and other appraisals such as Causal Agent are not being considered (which would lead to Pride), the agent's current emotion would correspond to Joy. The actual representation is the active appraisal frame: Suddenness=1.0, Goal Relevance=1.0, Outcome Probability=1.0, and Conduciveness=1.0.

3.5 Calculating Intensity

In addition to determining an appraisal as a point in a multi-dimensional space (or as a category), the system must also determine the *intensity*. Intensity is important because it summarizes the importance of the emotion, and thus indicates to what degree it should influence behavior. Emotions with low intensity are likely to be caused by less important stimuli than emotions with high intensity.

Overall our approach combines the numeric dimensions of the active appraisal frame to form a single numeric intensity value; since the categorical dimensions are non-numeric, they do not participate in the intensity calculation.

3.5.1 Criteria

There are many ways to produce an intensity value from a frame, and although there is little theory or empirical evidence to guide us, we define three general criteria for an intensity function:

1) Limited range: Intensity should map onto [0,1]. This is common to most existing theories.

2) No dominant appraisal: No single appraisal value should dominate the intensity function; each should contribute to the result but no single value should determine the result. This criterion eliminates a commonly used basis for combination: multiplication (e.g., Gratch & Marsella, 2004). One critical problem with multiplication is that if any dimension has a zero value, then the intensity will be zero, regardless of the other values.

3) Realization principle: Expected stimuli should be less intense than unexpected stimuli (Neal Reilly 2006). This is in contrast to Gratch & Marsella (2004) where intensity is maximized when the likelihood is 1.

3.5.2 The Intensity Function

To construct our intensity function, we begin with the last criterion. In our model, Likelihood most closely maps onto Outcome Probability (OP). However, rather than computing the change in Outcome Probability, we instead rely on the value of Discrepancy from Expectation (DE). These dimensions together imply a change in likelihood. If outcome probability and discrepancy from expectation are both high, then the intensity should be high since expected outcomes were not met. Similarly, if outcome probability and discrepancy are both low, then intensity should be high again, because something that was considered unlikely actually happened. If outcome probability and discrepancy have opposite values, then intensity should be low. (because either a likely stimulus occurred or an unlikely stimulus did not occur). This leads us to the first part of our function, which we call the *surprise factor*:

$$I = (1 - OP)(1 - DE) + (OP \cdot DE) \ldots$$

This function has low values when Outcome Probability and Discrepancy are at opposite ends of their ranges (because each product will be a combination of a low and high value), and high values when they are at the same end (because one of the products will be the combination of two high values). For example, if Outcome Probability = .9 and Discrepancy = .1, then *I* = .18. Similarly, if Outcome Probability = .9 and Discrepancy = .9, then *I* = .82.

To meet the first and second criteria, we notice that a simple function that allows each dimension to contribute is an average. A sum will not work because it would exceed the legal range as defined by the first criterion. To get magnitudes, we take the absolute values of those appraisals that can be negative. In general, one might expect that some dimensions contribute more than others do in the intensity calculation. In the absence of supporting data, however, we will assume all dimensions contribute equally. Thus, we normalize the dimensions with a [-1, 1] range[2].

We must now combine these two parts. Two obvious candidates are multiplication and averaging. We have chosen multiplication. An implication of this is that, if there is either no surprise or none of the other appraisals has any magnitude, the intensity will be zero. This does not violate our "no dominate appraisal" criterion, since it requires the influence of multiple appraisals in either case. It also entails a different interpretation of the "realization principle" than averaging would—multiplication ensures a completely expected outcome results in zero intensity, whereas averaging merely implies reduced intensity.

Thus, for the subset of appraisals we are considering in this thesis, we have:

$$I = [(1 - OP)(1 - DE) + (OP \cdot DE)] \cdot \frac{S + UP + \frac{|IP|}{2} + GR + \frac{|Cond|}{2} + \frac{|Crtl|}{2} + \frac{|P|}{2}}{num\_dims}$$

where OP=Outcome Probability, DE=Discrepancy from Expectation, S=Suddenness, UP=Unpredictability, IP=Intrinsic Pleasantness, GR=Goal Relevance,

[2] As described in Chapter 8, this normalization was removed in the revised system.

Cond=Conduciveness, Ctrl=Control, P=Power, and num_dims is the number of dimensions included in the average (7, if all dimensions have values).

In those cases where one or more values for appraisals in the averaging part of the equation are missing (as in our current simple choice reaction task example), the average is taken over the values that are present. If either Outcome Probability or Discrepancy from Expectation is missing, then the present value is multiplied by the averaging part (in this model, the Outcome Probability is always present in an active appraisal frame since there is always a prediction).

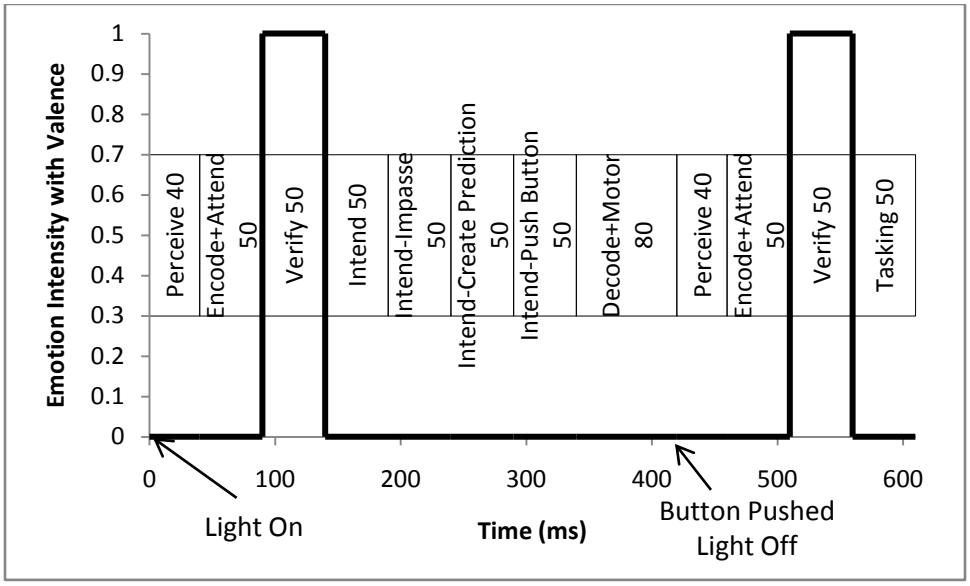


Figure 3.2: The task as split into PEACTIDM stages with the signed emotion intensity at each point in time.

3.5.3 Implications of the Intensity Function

The intensity function is biased so that some classes of emotions are inherently more (or less) intense than others. For example, the emotions that Scherer's theory would label as Boredom/Indifference are composed of low values for most dimensions combined with high outcome probability and low discrepancy, resulting in low intensity (see Table 2.2 for Scherer's mapping from appraisals to emotions). On the other hand, Scherer's Rage/Hot Anger emotions are composed of mostly high values, with high outcome probability and high discrepancy, resulting in high intensity. This is congruent

with many circumplex models of emotion (see section 8.3.3), which also propose different intensities for different emotions, suggesting a bridge between circumplex models and appraisal models.

3.6 Modeling the Task

Returning to our example, the intensity of the Joy following the light coming on is Outcome Probability multiplied by the average of Suddenness, Goal Relevance, and Conduciveness. Since these all have value 1, the intensity is 1. Figure 3.2 shows the entire task in terms of the PEACTIDM stages with the emotion intensity at each point in time.

Next, the agent verifies the prediction in the Comprehend step. Recall that the prediction was created before the task began, and it said that a light would come on. The prediction was accurate, so a value of 0 is generated for Discrepancy from Expectation. This causes the intensity of the emotion to drop to 0 because the surprise factor of the intensity is 0 (we might now call the emotion boredom).

Following Comprehend, the agent Intends to push the button. As described earlier, this causes the architecture to generate a prediction that the light will go off when the button is pressed, and it generates the command to push the button. The prediction replaces the previous prediction (that the light would come on) and has a new Outcome Probability associated with it (again, let's assume it is 1). This is followed by Decode and Motor with the result that the button is pushed and the light turns off. This change is Perceived, Encoded and Attended with appraisals generated as before, again resulting in a positive emotion with an intensity of 1. Comprehend confirms the prediction, causing the intensity to return to 0. Finally, Tasking marks the task structure as complete.

3.7 The Revised Task

When the world behaves as expected, there is very little to get excited about. Emotional reactions are often strongest when unexpected things occur. To explore this, we revised the task so that the light does not turn off when the button is pushed. How does this change the appraisals? The first part of the task (up to the pushing of the button) is exactly the same so that the Suddenness and Goal Relevance appraisals have values of

1, just like before. However, now when the button is pushed, nothing happens, so that when the stimulus (the light) is Attended to, Conduciveness is -1 because the stimulus is not on the path to the goal, as shown in Figure 3.3. The intensity of the emotion is still 1, but the valence is negative (because Conduciveness is negative). Our labeling function (section 5.1.1) calls this appraisal frame Displeasure. Comprehend determines that the prediction was inaccurate, resulting in a Discrepancy from Expectation value of 1. Thus, whereas before the intensity returned to 0 at this point, it now stays at 1, and thus the negative emotion persists (see Figure 3.3). We can only speculate at what would happen next, since the situation is presumably not covered by the task instructions; in our version, the agent still does Tasking and marks the task as complete.

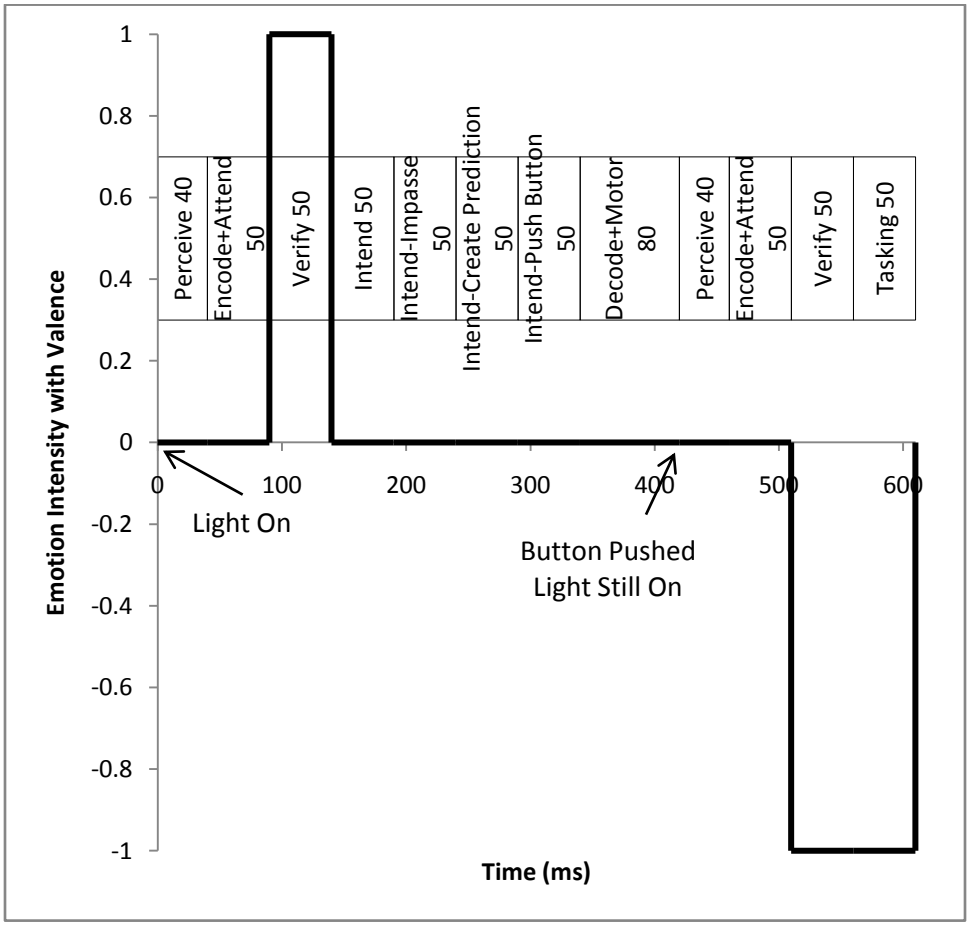


Figure 3.3: The revised task as split into PEACTIDM stages with the signed emotion intensity at each point in time.

3.8 A Brief Look at Human Data

Given that we are exploring this simple task for which human timing data exists, the question naturally arises: does it match the human data? To be clear, the goal of this thesis is not to match human data. Rather, it is to explore the integration of cognition and emotion, for which virtually no human data exists. But since we have the opportunity to explore the human data, we will do so briefly here.

The data reported by Newell (1990) states that the average reaction time from when the light comes on to when the button is pushed is 350 ms, with considerable variance. This corresponds to the first part of the task as described here. By looking at Table 3.1, we see that the model reported here takes 420 ms. However, the model reported here did not know how to directly implement the Intend function, and thus had to impasse and do the required subfunctions in sequence. Soar has a learning mechanism called *chunking* that can learn new rules based on the results generated during an impasse so that, in the future, the impasse can be avoided and the operator, in this case Intend, can be implemented directly. In this case, chunking learns rules that allow the agent to generate the push button command and create a prediction in parallel[3]. This results in the following timing:

Time (ms)	PEACTIDM Processing	Notes
0		Light on
40	Perceive	
90	Encode+Attend	
140	Comprehend	Verifies prediction
190	Intend	(Push button and Create prediction)
270	Decode+Motor	Light off

Table 3.3: Timing of first part of task after chunking.

After chunking, the task actually completes too fast. In Newell's (1990) original sketch of how Soar would perform this task, an extra "discrimination" operator was included after the verify operator in the Comprehend step. The purpose of this operator was to tell which light had actually come on so the proper button could be selected.

[3] We do not use chunking in the rest of the thesis. This is because the rest of the thesis is not concerned with human timing data, and ignoring chunking allows us to avoid dealing with certain chunking issues that may arise in more complex models.

Adding this to the model would push the timing up to 320 ms, which is considerably closer to the human data. However, in Soar, it was much more natural for this information to be generated by the Encode step. Since this does not depend on an operator, it does not take additional time.

Alternate timings for the steps that result in a closer fit are possible, but to date detailed human timing experiments have not been done in this version of Soar, so we do not have guidance in refining this model. Regardless, the takeaway point is that the timing data is not grossly off; it is still in the ballpark.

3.9 Discussion of the model

The emotional reaction of an agent to the task depends on at least two factors: to what extent the things occur as the agent has predicted them to, and what is at stake for the agent. In Figure 3.2, the agent has very brief reactions to the stimulus (in Soar, on the order of 50 milliseconds), which immediately go away when the agent realizes that the results are consistent with its expectations. This demonstrates how incrementally generated appraisal information leads to the emotion time courses. In Figure 3.3, when the outcome is unexpected, the agent's reaction is prolonged. Thus, even for a mundane task like pushing a button, emotional responses are possible. In the example, the appraisal values were extreme for demonstrative purposes, which would reflect a situation in which the consequences of the agent's actions are extremely important—such as the World Championship of button pushing, or if a large amount of money is riding on the agent's performance. One has only to watch TV game shows where the only action is choosing a box to open to see examples of extreme emotional responses for mundane actions. To emulate mundane button pushing, lower appraisal values would be used, which would result in little emotional reaction.

3.10 Summary

In this section, we demonstrated the integration of PEACTIDM and appraisals in our implementation. This included many details that go beyond PEACTIDM and appraisal, including value ranges for appraisals, active appraisal frames, and the

calculation of intensity. Finally, we touched on our model's relationship to human data and showed that, with learning, it is in the ballpark.

The next chapter describes the model in the context of a task that involves multiple actions over time. At the end of that section will be a discussion of some of the implications of the model which apply equally well to this simple model, but which the reader may find easier to appreciate in the more complex context. Hence, that discussion is delayed until then.

Chapter 4

A Non-Learning Model in a More Complex, Extended Task

In the previous chapter, we described the integration of PEACTIDM and appraisal theory in Soar in the context of a very simple task. In this section, we extend that model to a more complex (but still fairly simple) extended task that utilizes more appraisal dimensions. Unlike the previous task, this task may take an arbitrary number of PEACTIDM "cycles" to complete. This raises new issues, such as how previous emotions affect new emotions, and the role of Tasking when the ongoing task may be viewed as different subtasks. Addressing these issues will allow us to address qualitative questions such as, does the model produce coherent, useful behavior in the long term? Do the appraisals affect behavior and vice versa? Do appraisals have a reasonable (if not human-matching) time course? These and other questions will be addressed in the evaluation (Chapter 5).

For an ongoing task, we have chosen a simple Pacman-like domain called Eaters (Figure 4.1) that eliminates complexities of real-world perception and motor actions, while supporting tasks that although simple, allow for a range of appraisals and emotions. Eaters is a 2-D grid world in which the agent can move from square to square except where there is a wall. The agent can sense the contents of the cells immediately to its north, south, east and west. The agent's task is to move from its starting location to a specified goal location. This may not always be possible, in which case an intelligent agent should choose to give up so it can move on to other tasks. The task ends when the agent notices it has achieved the goal or when it gives up. Later in the thesis (Chapter 8) we will present a more complex model.

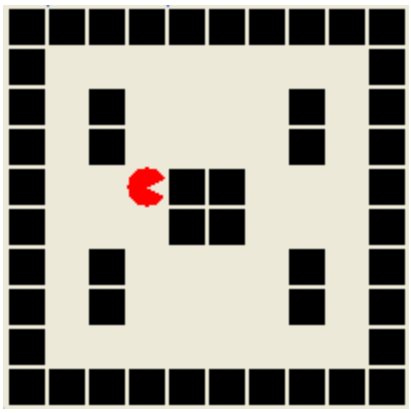


Figure 4.1: A screenshot of eaters.
The agent is the Pacman-like figure at location (3,4), walls are black cells, and open spaces are light-colored cells.

In terms of PEACTIDM, the agent will need to Perceive its surroundings, including information about what lies in each direction (e.g., walls, open spaces), create structures representing the encoded form of the input (e.g., some direction is passable and whether moving in that direction leads closer to the goal), Attend to one of the encoded structures, Comprehend that structure in terms of its current understanding of the situation (e.g., is the situation what the agent predicted), Intend an action if possible (e.g., if the Attended structure can be acted upon to get closer to the goal), and then perform the Intended action (via Decode and Motor). Tasking will play a role when the agent is stuck; for example, it may need to create a subtask to circumvent a wall, or to give up.

In appraisal theory terms, each choice point (e.g., what to Attend to, what to Intend, when to give up) will be guided by emotional information. Thus, the steps preceding these choice points must generate the appraisals that, directly or indirectly, influence the choices to be made.

What follows are the details of how each PEACTIDM function is implemented in this model, including how the appraisals fit in.

4.1 PEACTIDM in the Eaters Domain

This section describes how PEACTIDM as implemented in Soar is used to perform the Eaters task. Some aspects of these phases are domain-specific (e.g., the

stimuli and actions), but most of the core processing (Encode, Comprehend, Tasking) is general and taken directly from the previous model.

4.1.1 Perception and Encoding

Perception and Encoding generate structures that lead to relevance appraisals used by Attend to determine which stimulus to process. We do not directly model the Perceive function. The Eaters environment provides symbolic inputs to the Soar agent. Each direction (north, south, east and west) is considered a stimulus; thus, a separate structure is Encoded for each direction, which includes information such as whether the direction is passable, whether it is on the path to the goal or not, the distance to the goal, and whether the agent is making progress. The distance to the goal is an estimate based on Manhattan distance and may be incorrect if there are walls between the agent and the goal. If the agent is at a goal location, it will have a separate Encoded structure for the goal completion. The Encoded structure is fairly general—any task in which there is a path to the goal that can be blocked and where there is an estimate of distance to the goal can be Encoded in this way.

Figure 4.2 shows an example that will be used throughout the rest of this section. The goal is for the agent to reach location (7,4) (marked by the star) and the agent has moved from the west. The agent will have four encoded structures, one for each cardinal direction. The north, south and west structures will be marked as passable, directly off the path (since those directions will increase the distance to the goal), and at a distance of 4 from the goal. The east structure will be marked as impassable but directly on the path to the goal.

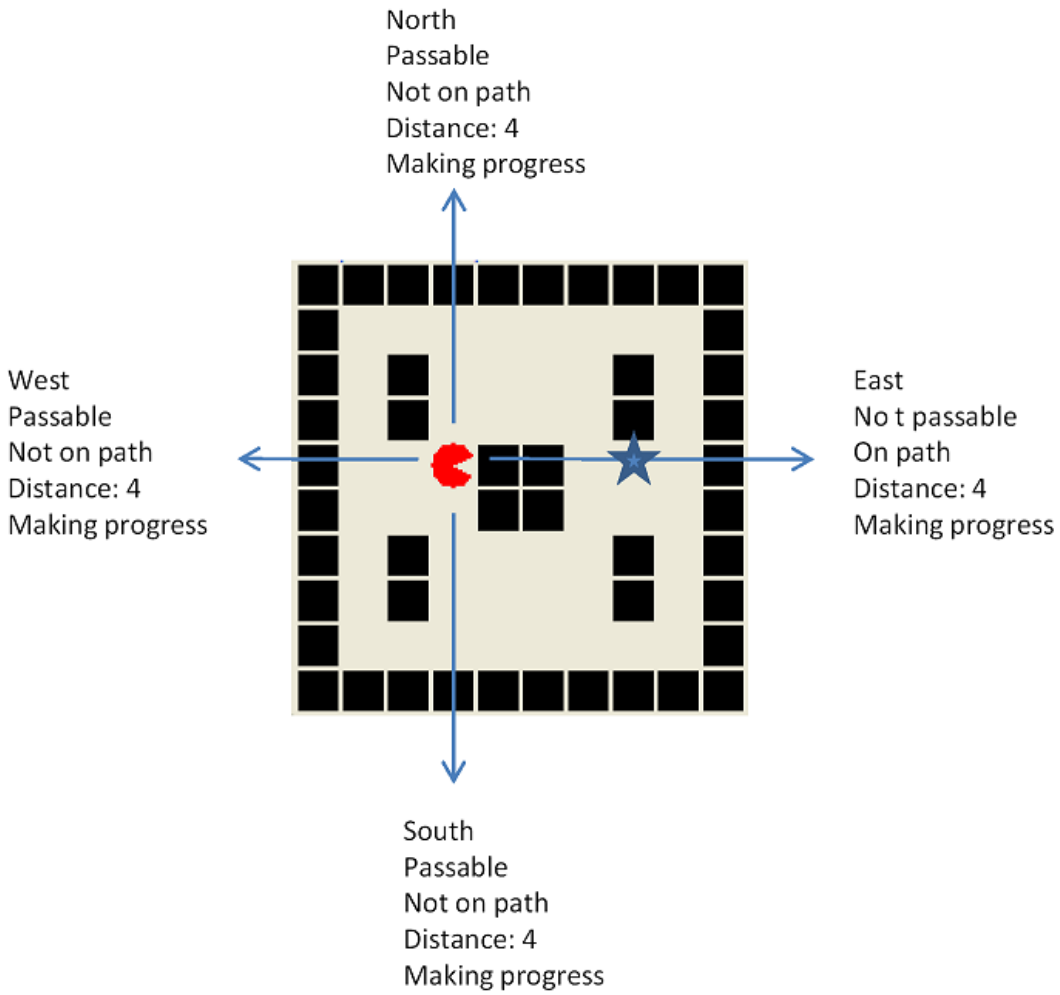


Figure 4.2: Encoded structures for each stimulus.
The star shows the goal location.

Relevance appraisals are generated directly from these Encoded structures. The north, south, and east stimuli have some Suddenness, whereas the west stimulus has no Suddenness (since the agent just came from there). In any environment, the agent will likely have some general expectations about what things to expect, and our agent expects there not to be many walls in the world. Thus, the north, south and west stimuli have low Unpredictability, but the east stimulus has a high Unpredicatability. Our agent is also averse to walls (since they only ever get in its way). Thus, it finds them Intrinsically Unpleasant giving the east stimulus a low Intrinsic Unpleasantness value. Finally, since the east direction is on the path to the goal, it is highly Goal Relevant, but the other

stimuli are not (Figure 4.3). Note that, in this model, only one goal or subgoal is active at a time, and thus Goal Relevance is computed with respect to that goal.

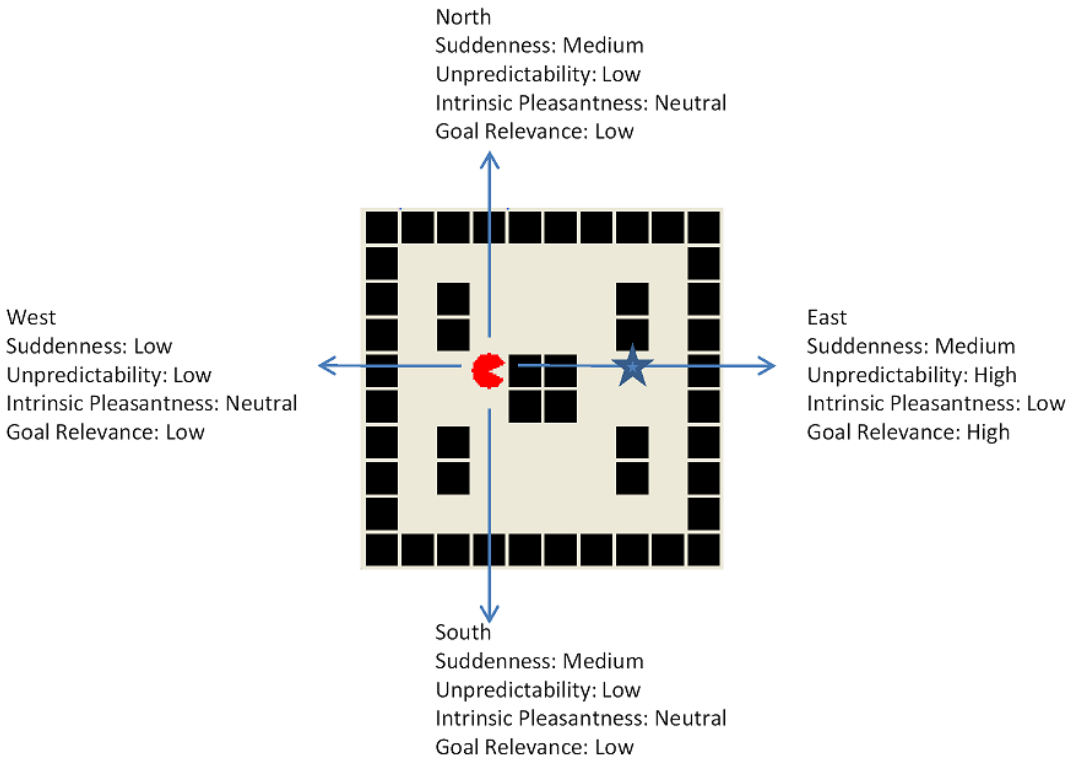


Figure 4.3: Pre-attentive appraisal frames for each encoded structure.

4.1.2 Attending

In general, the agent wants to make progress towards its goal, so stimuli that are Goal Relevant should given priority. However, Sudden or Unpredictable stimuli may also require attention, since these may be signals of danger or opportunity that needs to be dealt with. This is essentially an exploit versus explore tradeoff. Finally, stimuli that are intrinsically pleasant or unpleasant (independent of the current goal) may also deserve attention. In this model, each stimulus is appraised along the Suddenness, Unpredictability, Intrinsic Pleasantness, and Goal Relevance dimensions, determining the appraisal frame (Figure 3.1).

In this model, the selection of which stimulus is Attended to is a weighted random choice, with weights determined by the values of the appraisals just discussed. Since unusual stimuli are more likely to be worthy of Attention, as described above, appraisals with more extreme values lead to larger weights; that is, more interesting stimuli are more likely to be Attended to. Thus, the appraisals provide a task-independent language for knowledge that can influence control.

In our example, the north and south Attend proposals have moderate weights, whereas the west Attend proposal has a slightly lower weight (since its Suddenness is lower). The east Attend proposal has a higher weight because it is on the path to the goal, leading to an appraisal of Goal Relevance, and it has a wall, which is Intrinsically Unpleasant. Thus, the agent is most likely to Attend east.

4.1.3 Comprehension

Next, the agent performs the Comprehend function, which adds several additional appraisal values to the active frame (Figure 4.4). The agency of the stimulus is determined (in this model, "nature" is always the Causal Agent and "chance" is always the Causal Motive). The Conduciveness is also determined—if the stimulus direction is passable and on the path to the goal, it has high Conduciveness, whereas if it is off the path or blocked it has low Conduciveness. The Control and Power appraisals are also generated—if a stimulus direction is passable, Control and Power are rated high, whereas if the direction is impassable, Control and Power are low. While this domain is very simple, and thus the generation of these appraisal values is very simple, a more complex domain would potentially require arbitrary processing to determine values for any of these appraisals. We will not consider such extended processing here.

In our example, since the agent is Attending to the east stimulus, which is impassable but on the path to the goal, it will generate appraisals of low Conduciveness, low Power, and low Control (since it can't walk through walls). Causal Agency and Motive are "nature" and "chance", as noted above.

As in the previous model, the agent then Comprehends the stimulus by verifying it via comparison to the current prediction (as generated by the previous Intend) leading

to the generation of the Discrepancy from Expectation appraisal. If the stimulus is a match, then the Discrepancy from Expectation appraisal is low; if there is not a match, then the Discrepancy is high.

Unlike the previous model, once a stimulus has been verified, the agent performs another Comprehend step that determines if further processing is warranted. This gives the agent a chance to "back out" if it determines that processing should not proceed. That is, the agent answers the question, can additional processing of this stimulus lead to an action that helps me? The agent uses a heuristic called *dynamic difference reduction* to make this choice. Difference reduction (Newell, Shaw & Simon 1960) attempts to take internal processing steps to reduce the difference between the current state description and the goal state description. Dynamic difference reduction (Agre 1988) takes the steps in the world to avoid the need for increasing amounts of memory to track one's imaginary progress. Thus, difference reduction leads to plans whereas dynamic difference reduction leads to actions. In our model, if a stimulus can be acted upon (i.e., it is associated with a passable direction) and it does not lead directly away from the goal, then Comprehension is complete and the agent acts upon it (it does the Intend function). Otherwise, the agent chooses a second Comprehend operator, Ignore. Ignore marks the stimulus as processed and allows control to return to Attend, which will choose another stimulus to process from the remaining stimuli as above. This deactivates the appraisal frame for the Ignored stimulus.

In our example, the agent is Attending east, which is a wall. Comprehend will find a mismatch (since our simple model almost always predicts a passable route to the goal). This will trigger an appraisal of high Discrepancy from Expectation, which is added to the current frame. Since there is a wall, the agent cannot directly act upon the stimulus, so it then Ignores it. In fact, the agent is trapped by its goal in this case. As it Attends and Comprehends to each stimulus, it will find that the remaining stimuli lead away from the goal. Thus, Ignore will eliminate all of the remaining stimuli.

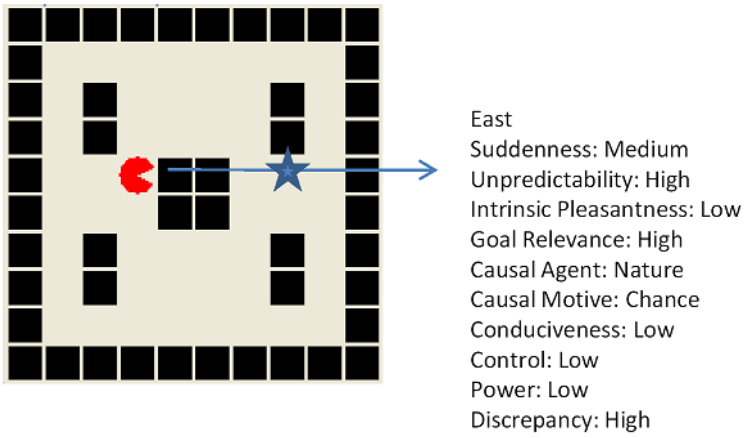


Figure 4.4: The agent attends East, making that appraisal frame active. The Comprehend function adds to this frame. The agent decides to Ignore this stimulus.

4.1.4 Tasking

When the agent has no options left, it is forced to engage in Tasking. This is an addition to the previous model which did not engage in Tasking during the task itself (only before the task began and at the very end). Generally speaking, Tasking is about managing goals (e.g., creating goals, giving up on goals, etc.). In this case, the agent creates a subtask to get around the blockage. In general, there are at least two types of goals. One type is abstract—the goal cannot be acted upon directly and must be broken down into more concrete components (perhaps many times) until it is in a form that can be directly acted upon. For example, the goal "Go to Work" is very abstract, and must be broken down to something that can be directly executed, such as "take a step". The other type is concrete—the goal can be acted upon directly. This is the form of goals in this model. When the agent temporarily retasks itself for the purpose of making progress on its original goal, we call this subtasking, and we call the new goal structure a subtask.

The goal that the agent cannot make progress on is to go to (7, 4). The reason that the agent is stuck on this goal is that its control knowledge and task formulation are too restrictive. Movement in any available direction will take it further from the goal, which violates its dynamic difference heuristic. In order to move around the blockage, it needs to temporarily get further away from the goal. Thus, the agent needs to retask and create

a goal that is less constraining, allowing it to get further from the main goal, but without violating its constraints in the new goal. The agent does this by defining the step it would ideally take—in this case, it would ideally move east to x=4. It sets this as its new subtask. That is, there is no constraint in the y (north-south) direction.

When an agent creates a subtask, it records information that gives it some idea of whether it is making progress or not. Specifically, it records the distance to the parent task (goal) at that time. It also tracks the minimum distance it has ever been to the goal upon entering a subtask. If the current distance to the goal is less than the minimum distance to the goal, then the subtask is considered a "good" subtask—that is, the agent knows that, even though it has to retask, it is making progress towards the goal. If the distance to the goal is not reduced, then the subtask is considered a "bad" subtask—that is, the agent cannot tell if it is actually making progress by retasking. The Encode function adds this good/bad subtask information to each Encoded structure, and this information influences some of the appraisals. In this model, the Conduciveness appraisal is more positive in good subtasks.

As alluded to above, once the agent has this new subtask, the Encoded stimuli are regenerated (since there is a different context for them now) and the agent can then re-Attend to the stimuli to see if any are now suitable. The agent can theoretically create an arbitrary number of nested subtasks this way, but for the current task it only needs one at a time (although it may create several in the course of completing the goal).

In our example, this is the agent's first subtask, so it defaults to a good subtask. The agent might still Attend to the east stimulus first and ignore it again, but when it Attends to, for example, the north stimulus, it will find that it is no longer directly off the path to the subtask. Instead it is now a sideways move (since it neither gets it closer to nor further away from x=4). Thus, the agent determines that this stimulus can be used for Intention processing (Figure 4.5).

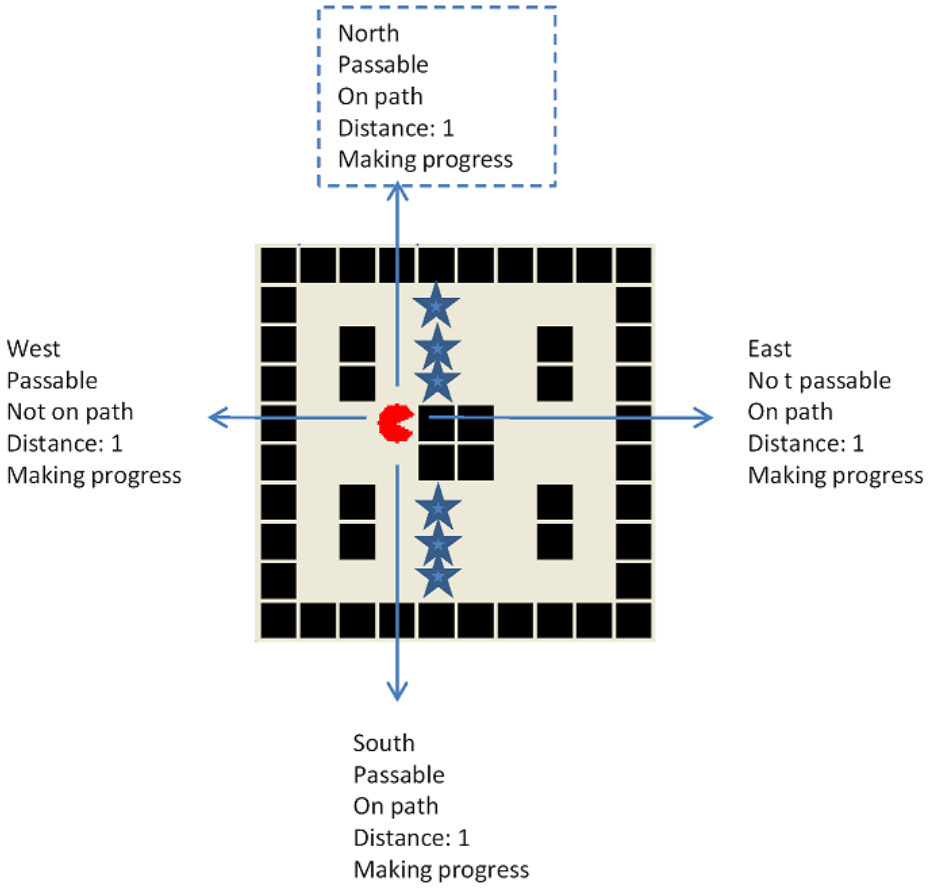


Figure 4.5: The agent creates a subtask to get around the blockage. The stars show the possible locations that would solve the subtask. This causes new encoded structures to be created. The agent Attends north.

4.1.5 Intending

Once the agent has found a stimulus it can act upon, it performs the Intend function, which is also implemented as a Soar operator. As in the previous model, Intend proposes moving in the direction of the stimulus. It also creates a new prediction structure—namely that the next stimulus direction will be passable and on the path to the goal (Figure 4.6) in this model, the agent is always optimistic in this way). If the agent is currently one step away from the goal, then it creates a goal achievement prediction. Along with the prediction, the agent also generates an Outcome Probability appraisal. As before, the Outcome Probability is tied to the prediction, and thus all appraisal frames in the situation that results from an Intend will inherit this same Outcome Probability (Figure 3.1).

In our example, Intend proposes moving north. The Intend operator sends a command to the environment to move north, and also creates a prediction. Since it is pursuing a subtask, the agent is less confident of its predictions, so it only rates the Outcome Probability of this prediction as moderate.

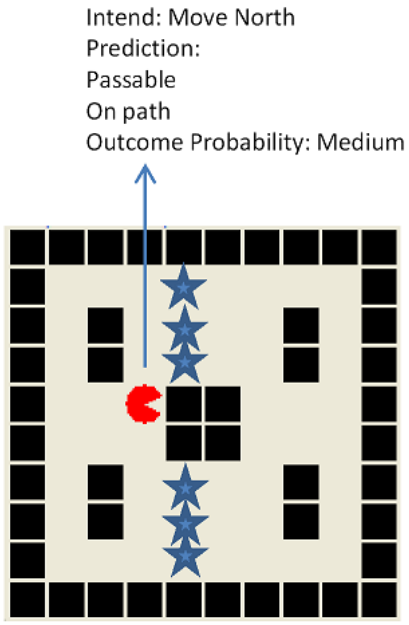


Figure 4.6: The agent Intends moving north.
It creates a prediction of the next stimulus it will see.

4.1.6 Decode and Motor

We do not directly model the Decode and Motor functions. The model uses Soar's standard method of communicating an action command to the simulated environment, which then executes it, leading to a new input state. For simplicity, in the model presented here, actions never fail (e.g., if the agent Attends to a wall, it will Ignore it instead of trying to move into it). However, our work on learning does allow action failures (see Chapter 7).

4.2 Emotion, Mood, and Feeling

In the previous model, we described how active appraisal frames become emotions. That is still true in this model. However, since the agent behaves over a long

period of time in this task, the question naturally arises, how do emotions affect each other over time? In this section we will introduce mood and feeling. The functional aspects of these will be discussed in section 4.3.

Recall that some existing computational models attempt to address the issue of how an emotion affects a succeeding one (see Chapter 6). Still, these models, and most theories, do not make an explicit distinction between emotion, mood and feelings; some only describe emotion (Hudlicka, 2004), some only describe emotion and mood (Gratch & Marsella, 2004) and some describe emotion, but mood only vaguely (Smith & Lazarus, 1990). One existing distinction made between emotion and mood is in terms of timescale: emotions are short-lived while moods tend to last longer (Rosenberg 1998). Some physiologically-oriented theories of emotion (Damasio 1994, 2003) distinguish between emotions and feelings: emotions have some impact on physiology, and the agent perceives or *feels* these changes, called the agent's feelings. That is, feelings are our perception of our emotions.

This distinction between emotion, mood and feeling is not universally accepted; indeed, what processes and phenomena are considered "emotional" is a subject of considerable debate. In our model, the specific labels are less important than the computational processes, structures and connections that make up the model as a whole. For example, Frijda et al. (1989) consider action tendencies to be part of emotion, whereas in our model we have action tendencies separate from emotion. Nevertheless, since the architecture supports the generation of action, and we have added the ability to generate emotion, mood, and feeling, the mechanisms are in place to allow an integration of these with action. Indeed, action is partially influenced by feeling in the present model (see section 4.3). That these phenomena are inextricably bound is not debated; how we choose label them is an expository convenience.

In our model, we maintain a distinction between emotion and feeling, and also introduce mood. Emotion is the currently-active appraisal frame. In our model, we use a simple model of mood, where mood is a weighted average-like aggregation over past emotions computed at the individual appraisal level, so that mood is represented as an appraisal frame. This initial model of mood captures some of the time course and

interactions among emotions, while ignoring many of the complexities of a more complete model of mood. Feeling is the combination of emotion and mood, represented as an appraisal frame, augmented by an intensity. Thus, in the previous model, what we reported as the agent's emotion with intensity (e.g., Joy, 1.0) is actually the agent's feeling and feeling intensity. Since feeling is represented using an appraisal frame, the intensity calculation we proposed previously (section 3.5) still applies.

The remainder of this section contains a lot of details that, while important to understanding exactly how the system works, are irrelevant to the big picture, so the reader should feel free to skip ahead to section 4.3 on page 58.

Figure 4.7 shows how the agent generates an appraisal frame (its emotion), which interacts with another appraisal frame (its mood) to generate its perceived appraisal frame (its feeling). We call these appraisal frames because their structure is a collection of appraisal dimensions, not because the agent, via an appraisal process, directly sets the contents of them (the agent only directly sets the contents of the emotion frame). When necessary, we will distinguish among them by referring to the emotion frame, the mood frame, or the feeling frame. This contrasts with most other theories in which emotion, mood, and feeling are not distinguished by separate structures.

The representation of feeling as an appraisal frame is most likely a simplification because feelings are the perception of the physiological reactions to the combination of mood and emotion (Damasio 1994; 2003). Nonetheless, whatever structure is produced will be the basis for intensity, and the analysis we develop below should apply to that structure. Non-computational ideas regarding such structure have been proposed (Lambie & Marcel 2002).

Given a feeling frame, the system calculates the intensity of that feeling (using the method described in section 3.5). Intensity gives the agent an indication of how important a feeling is, and thus helps determine to what extent the feeling should influence behavior.

Thus, the remaining questions are, how is the mood appraisal frame generated, and how does the emotion appraisal frame combine with the mood appraisal frame to

produce the feeling appraisal frame? In the creation of the theory, we have tried to rely on existing work and data. However, for the level of detail required in a computational model, such prior work is limited. Thus, we are faced with numerous decisions where there is little or no guidance from the literature. In these cases, we have tried to choose the simplest alternative (recognizing that "simplest" can be a subjective concept); that is, we are applying Occam's Razor. Our long-term strategy (beyond this thesis) is to see where these simple assumptions fall short, which will indicate where additional complexity is required. Thus, the theory we present is likely oversimplified, but it provides a starting point for future work.

We will begin with our model of mood, and then describe how it combines with emotion to generate feeling.

4.2.1 Mood

In our model, emotion is based on the agent's appraisal of the current situation independent of any historical context. To avoid wild fluctuations in feeling, historical context is necessary, but this context should be biased toward those evaluations that are temporally relevant. Mood provides this historical context of recent emotions. Thus, we make the simple assumption that the mood combines with the current emotion to form the feeling that the agent perceives; more complex models are possible, of course.

To the extent that mood is physiological in nature, there are some phenomena that can guide our model. In the undoing effect (Fredrickson & Levenson, 1998), physiological changes due to negative emotions return to baseline (the natural state for some positive emotions) more quickly when followed by a positive emotion. One possible interpretation of this is that the mood "chases" the emotion (i.e. the mood tries to change to match the state defined by the emotion), but will still decay on its own if left alone.

Mood starts out neutral (i.e. all zero values). To model the influence of emotion on mood, the mood "moves" towards the emotion each time step. In the current model, we have adopted a simple approach where the mood moves x% (our current experimental value is 10%) of the distance along each dimension towards the emotion in each cycle.

Additionally, the system decays mood by y% (experimental value is 1%) each cycle. Thus, each emotion influences mood for a theoretically infinite amount of time, but the magnitude of the influence decreases exponentially with time. Therefore, if there were no influence of emotion, mood would eventually become neutral. This model is summarized in Figure 4.7.

4.2.2 Combining Mood and Emotion to form Feeling

In general, the relationship between appraisal frames may be complex with interactions among multiple dimensions. However, we have no reason to assume this, so instead we simpler assumption that each appraisal dimension in a frame influences only the corresponding dimension in the other frame.

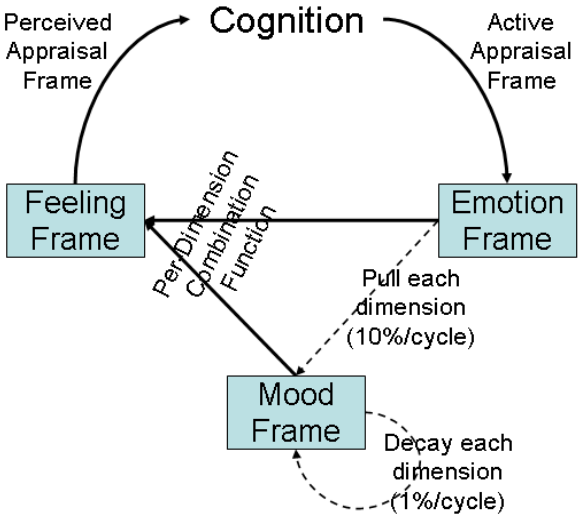


Figure 4.7: An emotion frame influences and combines with the mood frame to produce the feeling frame, which is perceived by the agent.

Before we can discuss how mood and emotion combine to create feelings, we must discuss the nature of the appraisal dimensions and their values that make up the frames.

4.2.3 Value Ranges for Categorical Appraisals

The value ranges for each appraisal dimension was described in section 3.1. However, we need to return to the issue of categorical representation: how is it that different categorical values can be combined across frames?

To address this issue, we use a numerical representation of the categorical appraisals for the purposes of combination. Causal Agent and Causal Motive can each take on three values: Self, Other, Nature; and Intentional, Chance, Negligence, respectively. Our approach is to convert these categorical values into mutually exclusive features, each with its own numeric value in the range [0, 1]. Thus, the original Causal Agent feature is expanded into three features: Causal-Agent-Self, Causal-Agent-Other, and Causal-Agent-Nature. For the emotion frame, the selected value gets 1 and the others get 0. For example, if the value of Causal Agent is nature, then the dimension Causal-Agent-Nature gets a value of 1 while Causal-Agent-Self and Causal-Agent-Other get 0. The values for these dimensions are now numeric and are treated like other numeric values so that the mood tracks recent historical values for these dimensions. The feeling value is then the combination of these dimensions from the frames, just like the other dimensions. However, after combination, multiple categorical values can be non-zero, representing confusion about which is the true value. In these cases, the agent perceives the categorical value of the dimension with the highest numeric value. Thus, if Causal-Agent-Self = .4, Causal-Agent-Other = .7, and Causal-Agent-Nature = .2, the agent would perceive Causal Agent = Other.

4.2.4 Criteria for the Combination Function

There are many options for combining the values of mood and emotion to produce a feeling; we introduce several criteria below that such a combination function should meet. Simple combination functions such as averaging or multiplication have been shown to be inadequate, as our criteria will illustrate. Existing work (Neal Reilly 1996, 2006) has already provided some relevant criteria; however, that work has been done at the more abstract level of emotions of the same kind (e.g. Joy .3 and Joy .2). Our theory is defined at a lower level, that of individual appraisal dimensions and their "intensity" (e.g. Suddenness .3 and Suddenness .2). However, the criteria defined for these higher-level models still apply at the lower level, because the criteria are about how to combine intensities of the same kind, and are agnostic as to the kinds are.

We make the simplifying assumption that the dimensions are independent, so our combination function takes as input a particular dimension from the mood and emotion

frames to produce the corresponding dimension of the feeling frame. This function is applied to each dimension of the frames.

We begin by noting that we want to avoid a large range of inputs from mapping onto a small range of outputs because then the agent will not be able to distinguish between those inputs, and thus will not be able to form diverse responses. This criterion is subjective.

1) Distinguishability of inputs: Large input ranges should have large output ranges. Capping of extreme values may be necessary, but it should have minimal impact.

Next, we consider constraints from prior work: when combining values of the same sign, the result should be further from zero than the input with the largest magnitude, but less than or equal to the sum of the inputs (Neal Reilly 1996, 2006). The intuition is that the values should build on each other, but the combination should not be more than the parts. For example, if the mood's Suddenness value is .3 and the emotion's Suddenness value is .5, the feeling's Suddenness value should be at least .5 but no more than .8.

For values of opposite signs, the result should be closer to zero than the maximum magnitude, but be at least the sum of the inputs. Furthermore, the result should be further from zero than the sum of the results. The intuition is that the smaller value is dragging down the larger value, but the amount of the reduction should be no more than the magnitude of the smaller value. For example, if mood's Conduciveness is .3 and emotion's Conduciveness is -.5, the result should be between -.5 and -.2.

We can state the above by defining the combining function *C*, which has inputs $v_{emotion}$ and v_{mood}:

2) Limited range: $C(v_{emotion}, v_{mood})$ should be between the input with the maximum magnitude and the sum of the inputs.

Another issue is that, if possible, the value should not go out of scale. This can happen with middle values combined with a strict sum (e.g. .6 and .6). Values can always be capped, but capping middle values means the agent will be unable to distinguish among a large set of possible inputs, which violates our first criterion. Thus, our next

criterion is that the combination should not be linear (Neal Reilly, 2006). While $C(.5, .5)$ should be much less than 1, $C(.1, .1)$ can be very close to .2. The intuition is that low-intensity stimuli can result in a moderate intensity reaction, but moderate-intensity stimuli should not result in extreme intensity reactions. That is:

3) Non-linear: For small inputs, C is nearly additive, but for large inputs, C is closer to a max. Put another way, for small values the derivative of C can be close to 1, but for large values, the derivative of C should be closer to 0.

We also identify several properties that enforce symmetry on the function. These properties do not result from any intuition or data, but rather represent reasonable first guesses given the lack of information. That is, these are default assumptions and not hard constraints. We would be satisfied with a theory that violated these criteria so long as the theory recognized the implications of the bias. For example, here may be some basis for a positivity bias (Diener & Diener 1996), but it is not clear whether such a bias belongs in the combination function or in the processes that generate the emotion frame.

4) Symmetry around 0: $C(x, 0) = C(0, x) = x$. If the mood or emotion input is 0, then the other input dominates. If they are both zero, then the result should be zero.

5) Symmetry of opposite values: $C(x, -x) = 0$. The mood and emotion can cancel each other out.

6) Symmetry of all values: $C(x, y) = C(y, x)$. The mood and emotion have equal influence on the feeling.

4.2.5 The Combination Function

As a starting point, we will use Neal Reilly's (2006) proposed function for combining intensity values of the same kind, and then modifying it as necessary to meet our criteria:

$$I = 0.1 \cdot \log_2 \sum_{em} 2^{10 \cdot em}$$

This function was designed to deal only with positive values. For most of those values, the function meets criterion 2 (limited range) and 3 (non-linear). The log combination ensures that the result is at least the max value, but no more than the sum.

Further, the derivative of the log is near 1 for small values, but decreases for larger values. For example, $I(.1, .1) = .2$, but $I(.5, .5) = .6$.

Perhaps surprisingly, the function fails criterion 2 (limited range) at the lower extreme ($I(0, .1) = .15$), 4 (symmetry around 0; $I(0, .1) = .15$). The function does fulfill criterion 6 (symmetry of all values) for positive values. Criterion 5 (symmetry of opposite values) does not really apply since the function does not deal with negative values.

The problems with this function can be fixed. To deal with negative values (criterion 5), we introduce a Sign function and absolute values. The absolute values allow us to work with the magnitudes of the inputs, while the Sign function allows us to restore the signs that were removed by the absolute values. To do this, we break the function into two parts: the sum part and the log part. The sum part treats the exponent as a magnitude, but applies the original sign before including the value in the sum (see function below).

To center the function at 0 (criterion 4), we recognize that we need to end up taking the log of 1 (to get 0). If each input is 0, then the result of the exponent will be 1, and thus the sum part will be 2. To fix this, we subtract 1 from each magnitude of the sum (so the sum will be 0 for zero-valued inputs), and then add the Sign of the sum to the sum before taking the log (to maintain symmetry).

We originally chose $b=e$ instead of 2 because the resulting values are less extreme near the edges of the input range, which helps meet criterion 1 (distinguishability of inputs). However, this function still fails criterion 1. The log scale of the function causes the result of an extreme input value and nearly any other input value of opposite sign to fall into a very narrow range. For example, $C(.9, -.1) = .89998$, whereas $C(.9, -.5) = .89816$—nearly the same value. To fix this, we introduce a piecewise function that varies b depending on the inputs. If the signs are equal, then $b=e$. If the signs are opposite, then $b=1.1$, which spreads out the resulting values. For example, $C(.9, -.1) = -.85453$, whereas $C(.9, -.5) = .58561$.

The final function is shown below. A complete example showing mood and emotion frames combining to form a feeling frame is shown in
Table 4.1.

$$C(v_{mood}, v_{emotion}) = 0.1 \cdot Sign(S) \cdot \log_b(|S + Sign(S)|)$$

where $$S = \sum_{v=v_{mood}, v_{emotion}} \left(Sign(v) \cdot \left(b^{10 \cdot |v|} - 1\right)\right)$$

and $$Sign(v) = \begin{cases} 1 \text{ if } v \geq 0 \\ -1 \text{ else} \end{cases}$$

and $$b = \begin{cases} e \text{ if } Sign(v_{mood}) = Sign(v_{emotion}) \\ 1.1 \text{ else} \end{cases}$$

If $C(v_{mood}, v_{emotion}) > 1$ then $C(v_{mood}, v_{emotion}) = 1$

If $C(v_{mood}, v_{emotion}) < -1$ then $C(v_{mood}, v_{emotion}) = -1$

4.2.6 Discussion of the Combination Function

The combination function, together with the intensity function we presented earlier, can sometimes lead to unexpected results. Even though the combination function has a building effect (i.e. if the inputs have the same sign, the magnitude of the result will be at least as large as the magnitude of the largest input), this will not necessarily result in a higher the intensity for the feeling. Given the way Outcome Probability and Discrepancy from Expectation influence intensity via the surprise factor, even if both of those values go up, the intensity may actually go down. For example, suppose the Discrepancy and Outcome Probability for the feeling are both .1 (and assume all other dimensions were 1.0). This would lead to an intensity of .82. However, if both of these dimensions then increased to .2, the intensity would fall to .68.

Unlike other models (Neal Reilly, 1996; Gratch & Marsella, 2004; Hudlicka, 2004), the mood and feeling processes do not combine emotions; they combine individual appraisals. This could lead to unexpected feelings. For example, an emotion best described as elation-joy combined with a mood best described as anxiety-worry can result in a feeling best described as displeasure-disgust. This is an interesting prediction of the model that we have not yet investigated.

Given the lack of relevant data on which to base our theory, rather than present comparative results, we instead demonstrate the system's behavior. First, we give a complete example showing the output of the combination and intensity functions and

discuss its consequences. Then we show actual feeling intensity data from the Eaters domain that demonstrates the realization principle.

Table 4.1 shows a complete example of mood and emotion frames combining to create a feeling frame, along with the intensity of each frame. While the agent only perceives the intensity of the feeling frame, it can be useful to generate intensities for the other frames to aid our understanding of the system.

	Mood	Emotion	Feeling
Suddenness [0,1]	.235	0	.235
Unpredictability [0,1]	.400	.250	.419
Intrinsic-pleasantness [-1,1]	-.235	0	-.235
Goal-relevance [0,1]	.222	.750	.750
Causal-agent (self) [0,1]	0	0	0
Causal-agent (other) [0,1]	0	0	0
Causal-agent (nature) [0,1]	.660	1	1
Causal-motive (intentional) [0,1]	0	0	0
Causal-motive (chance) [0,1]	.660	1	1
Causal-motive (negligence) [0,1]	0	0	0
Outcome-probability [0,1]	.516	.750	.759
Discrepancy [0,1]	.326	.250	.362
Conduciveness [-1,1]	-.269	.500	.290
Control [-1,1]	-.141	.500	.402
Power [-1,1]	-.141	.500	.402
Label	anx-wor	ela-joy	ela-joy
Intensity	.088	.094	.127

Table 4.1: An example combination of a mood and emotion frame to form a feeling frame. Approximate linguistic labels provided based on Scherer's (2001) modal emotions.

Figure 4.8 shows feeling intensity data excerpted from an agent in the Eaters domain. As the figure shows, feeling intensity is maximized when the agent first realizes that it will achieve its task, and is less when the agent actually achieves the task. This is because going into the state where the realization occurs, the agent has a prediction which assumes that the task completion is not imminent with some moderate Outcome Probability. The realization that task completion is indeed imminent violates this expectation. Thus, Outcome Probability was at least moderate, and Discrepancy from Expectation was high, leading to a higher intensity (assuming no major changes in the other appraisals). Following this, the agent now predicts that the task will be

accomplished with high probability, so when it is in fact accomplished, Outcome Probability is high and Discrepancy is low, causing the intensity to be lower.

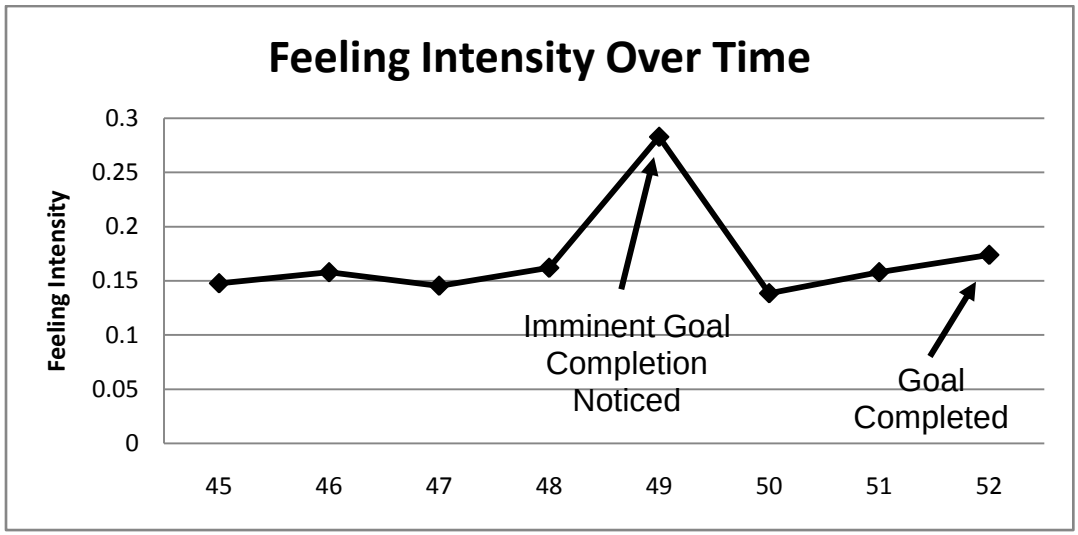


Figure 4.8: Feeling intensity is maximized when the agent realizes the task will be completed, as opposed to when it actually completes.

4.3 The Influence of Emotion, Mood and Feeling upon Behavior

Feeling adds knowledge to the state representation in a task-independent format that combines representations of current (emotion) and past (mood) situations, and thus is more general than emotion or mood alone. Feeling can be used to guide control, and thus it can influence behavior. Task-dependent representations can still influence behavior both directly (as in how the agent might choose to cope with its feelings in a particular domain) and indirectly (in that appraisals can be generated from task-dependent representations). Emotion theories describe a number of influences of emotion, mood, and feeling, including effects on cognitive processing (Forgas 1999) and coping (Gross & John 2003), and integration with action tendencies (Frijda et al., 1989). Our current approach is very simple, included to demonstrate the possibility of feelings influencing behavior and focusing on one aspect of coping: coping by giving up on goals.

Most AI systems, when faced with a difficult or impossible task, have no way to recognize that they should give up and will work on the problem until all resources are

exhausted. By providing emotional feedback, our model allows the agent to detect that it is not making progress towards the goal, and thus it can choose to discard that goal (possibly so it can move on to another goal or stop wasting resources). This behavior could be accomplished without emotions, moods, and feelings, but they provide a natural way to achieve this.

In our model, when the agent fails to make direct progress, it will form a subtask. While pursuing a subtask, the agent can choose to give up if its current feeling of Conduciveness is negative. Giving up is another form of Tasking—it removes the current goal. As this feeling intensity increases, the agent is exponentially more likely to give up. Mood plays a role here by tempering or enhancing the current emotion. Thus, if things are going well (mood is positive) but the agent experiences a momentary setback (emotion is negative), the overall feeling intensity will be lower, making giving up less likely. If things have been going poorly, however, the setback will build on that, resulting in a more intense negative feeling, making giving up more likely. The option to give up is in competition with other activities in the subtask, specifically attending to possible directions in which it can move. That is, the agent still makes a weighted random choice, with giving up being an option whose weight is exponential in the magnitude of the negative feeling intensity. As the agent eliminates more of its Attend options (by Attending to and then Ignoring them), it becomes more likely to give up (since there is less competition from other Attend proposals).

While the current model only has this single direct influence of feelings on behavior, each appraisal of each stimulus has an indirect influence. As described above, at the Attend stage, the pre-attentive appraisals influence where attention is focused next. Furthermore, past appraisals influence the current feeling via mood, and thus indirectly influence the agent's decision to give up or not.

4.4 Summary

To summarize, the integrated model described in Chapter 3 extends to an extended task such as the one described in this chapter with more appraisal dimensions involved. As before, the structures that many of the PEACTIDM steps use and generate

are based on appraisals. In this model, the Perceive and Encode step generates the Suddenness, Unpredictability, Intrinsic Pleasantness and Goal Relevance. Attend uses these to choose a stimulus to process further. Attending also enables the generation of other appraisals, including Conduciveness, Causal Agent, Causal Motive, Control, and Power. The Comprehend step is implemented as a verification operator (which generates Discrepancy from Expectation) and an ignore operator. Tasking allows for the generation of subtasks when there are no useful actions to take (that is, all other stimuli have been Ignored). Intend takes the action associated with the currently-attended encoded structure and creates a prediction of the outcome of that action, as well as the Outcome Probability of it. When pursuing a subtask, the agent has the opportunity to give up (another Tasking operator). This combination of appraisal and PEACTIDM leads naturally to sequential constraints on appraisal generation.

The probability of giving up is influenced directly by the feeling's Conduciveness dimension, and also indirectly by all the other numeric appraisals via the intensity of the current feeling. Feelings are determined by combining mood and emotion, with mood being influenced by emotion.

The model as presented does not use the Unfamiliarity, Urgency, Adjustment, Internal Standards, or External Standards appraisal dimensions from Scherer's theory. These were not critical for this domain, and adding these to our architecture is future work.

Finally, we want to note that although the model is implemented in Soar using Scherer's appraisal theory, the underlying theory is intended to be general. That is, we have not intentionally introduced any constraints that would prevent this theory from being implemented, for example, in ACT-R using a different appraisal theory (as discussed in section 2.1.4.1).

The next chapter describes experiments intended to evaluate this model.

Chapter 5

Evaluation of the Non-Learning Model

What kind of evaluation is appropriate for this model? Clearly, given the computational nature of the system, it is possible to generate quantitative results. However, given the lack of human data or existing systems to compare to, these results can only be used to support claims about the system itself, as opposed to a comparison.

First we consider Picard's (1997) properties that an emotional system should have:

1. Emotional behavior: System has behavior that appears to arise from emotions.
2. Fast primary emotions: System has fast "primary" emotional responses to certain inputs.
3. Cognitively generated emotions: System can generate emotions, by reasoning about situations, especially as they concern its goals, standards, preferences, and expectations.
4. Emotional experience: System can have an emotional experience, specifically cognitive and physiological awareness and subjective feelings.
5. Body-mind interactions: System's emotions interact with other processes such as memory, perception, decision making, learning, physiology, etc.

We begin with 3 (cognitively generated emotions). The system has this property as it uses cognitively generated appraisals as the basis for its emotions. Similarly, the system exhibits 2 (fast primary emotions) because the system generates appraisals beginning at the Perception and Encoding phases, and those become active at the Attend phase. While some have argued that appraisals are "too cognitive," and thus can't be

used to generate fast emotional responses (Zajonc, 1984), Soar naturally supports this fast appraisal generation, so long as no significant inference is required (Marsella & Gratch, in press). Indeed, one implication of the Scherer (2001) theory is that the relevance appraisals (suddenness, unfamiliarity, unpredictability, intrinsic pleasantness, goal relevance) are generated very early, and our system reflects that. Moreover, as soon as the appraisal frame becomes active, the appraisals become the emotion. Then, as further processing generates more appraisals, these are added to the emotion. In the case that additional processing is necessary, Soar can learn to speed that processing via its chunking mechanism, as discussed in section 3.8.

In terms of 4 (emotional experience), the system has some emotional experience but it is incomplete. The system is cognitively aware of its emotional state (the appraisals and the resulting feeling are available in Soar's working memory). Also, the feelings are subjective in the sense that the agent can, in principle, interpret them however it sees fit. While we did not explore this here, there is nothing that prevents cultural knowledge from being added that would allow the agent to generate labels for or other interpretations of the feeling frame the system generates. However, in the current implementations, it has only a trivial physiological system.

For 5 (mind-body interactions), emotions can influence decision making, in that the agent can decide to give up when its emotional state is bad. We evaluate this quantitatively in the context of coherent behavior below. In Chapter 7, we describe an extension of this system that learns as well. However, we have not yet explored connections to memories, perception, physiology, or a host of other areas that could be influenced by emotion.

The remaining criterion, 1, is whether the agent exhibits emotional behavior. We will explore this quantitatively below.

Picard's list can be extended with additional requirements. First, while we have described how the model works at the micro level, we have not yet demonstrated that it actually produces useful, purposeful behavior. Does it even finish the task? If not, does its emotional state justify the failure? Furthermore, there are several implications that

should be explored. For example, if an agent's feelings are determined by the available stimuli, then different environments should lead to different feelings. Additionally, even in environments where the distance to the goal is the same, since Attend takes information about the situation (in a task-independent representation) into account (e.g., Suddenness), different environments should result in different amounts of time to completion. We also claimed in the last section that feelings should impact behavior, both directly and indirectly. Thus, we suggest that there should be a loop: behavior influences feelings, which influence behavior.

We will show results that suggest the model meets the additional requirements described above (summarized here):

6. The model works and produces useful, purposeful behavior.
7. Different environments lead to differences in behavior, including:
    a. Different time courses
    b. Different feeling profiles
8. In a given environment where the agent has choices, these choices impact feelings and thus the agent's success.

As discussed earlier, for simplicity, we used a non-human agent in the synthetic Eaters environment. Thus, while we present time course data, these data should not be mapped onto real time for comparison to humans given the simplicity of the Eaters environment, sensors, and effectors.

5.1 Methodology

To evaluate the agent, we used several different mazes in the Eaters domain with a specific goal location in each. In each maze, the distance from the start to the goal was 44 moves (except for the last maze, in which it was impossible to reach the goal). Our aim in designing these mazes was to place the agent in progressively more difficult situations to demonstrate the properties listed above. In the first maze (Figure 5.1), the agent did not have to ever retask to reach the goal, and there were no distracting stimuli; that is, it could not see any walls on its way to the goal. The second maze (Figure 5.2) is exactly the same as the first except that the path to the goal is lined with walls (and hence

distractions). Thus, even though there are fewer possible moves, there are just as many Attend opportunities, and they are actually more interesting (hence, distracting). The third maze (Figure 5.3) is very similar to the second, except that there is a kink in the path that requires a brief retasking to maneuver around. This is because the agent has no direct way of making progress when it reaches the kink—if it moves north, it will be further from the goal, and it can't move east because of the wall. Thus, retasking allows it to temporarily move further from its original goal. The fourth maze (Figure 5.4) contains twists and turns such that four subtasks are required to reach the goal. In the fifth maze (Figure 5.5), it is not possible to reach the goal.

Figure 5.1: An Eaters maze without any distractions.

Figure 5.2: An Eaters maze with distractions.

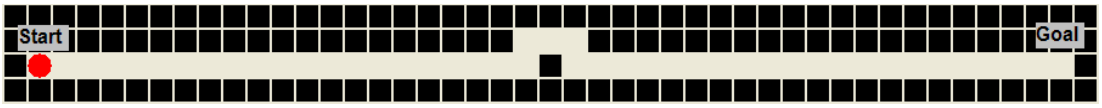


Figure 5.3: An Eaters maze with distractions and one subtask.

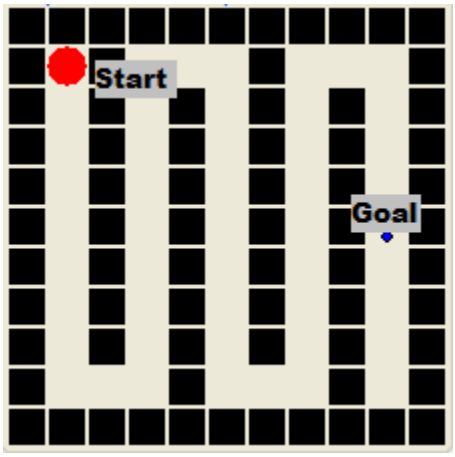


Figure 5.4: An Eaters maze with distractions and multiple subtasks.

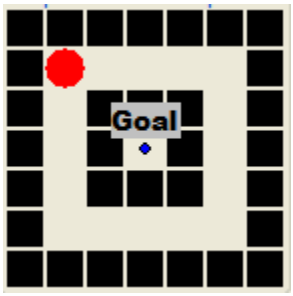


Figure 5.5: An Eaters maze that cannot be successfully completed.

5.1.1 Labeling Appraisal Frames

While the agent does not use linguistic labels to determine its behavior, we found such a labeling function is useful in analyzing the agent's behavior (indeed, we use it in the results reported here). The labeling function is based on the Manhattan distance between the agent's appraisal frame and the modal emotions defined by Scherer (see Table 2.2). Since some modal emotions have many unspecified values (which are treated as distance 0), some emotions are frequently closer to the feeling frame than others, even when their specified appraisal values are not good matches. Elation/Joy is one such emotion (it has open values for Intrinsic Pleasantness, Discrepancy from Expectation, Control and Power). To compensate for this, we only considered modal emotions that have a Conduciveness with the same sign (or an open Conduciveness). In other words, we divided the emotions into positive and negative emotions based on Conduciveness,

and ensured that only labels with the same valence as the frame could be applied. Thus, it is not possible for a feeling with negative Conduciveness to be labeled as Elation/Joy.

An unusual case in the labeling function is the Displeasure-Disgust label: Scherer defines it in terms of Intrinsic Pleasantness rather than in terms of Conduciveness (see Table 2.2), so we split instances of these into positive and negative, as defined by whether Conduciveness was positive or negative. Thus, positive Displeasure-Disgust is when that label most closely matches the current feeling, but Conduciveness is positive. This can occur when the agent must do something it dislikes, but is necessary to make progress in the task. Real-life examples might be washing the dishes or cleaning a toilet.

5.2 Results

In the first two mazes, the agent will never give up, since it never has to retask. However, we anticipate that the distractions from the walls in the second maze will make it take significantly longer to complete than the first, and that the agent will experience more negative emotions as a result. In the last three mazes, retasking is required and thus the agent can fail. In the third and fourth mazes, the addition of the subtasks require extra processing that could cause the agent to take longer to complete the mazes. Moreover, in the fourth maze, the agent is likely to give up before achieving the goal because of it detects it is not making progress. We expect that the agent will alway give up on the fifth maze because it is impossible to solve. We expect this to take less time than the fourth maze, because in the fourth maze the agent is always making progress, whereas in the fifth maze, after the first subtask, the agent detects that it is not making progress, which should lead the agent to feel worse and hence give up sooner.

Figure 5.6 shows the time course of behavior in the different mazes, as well as the success rate in each maze. As we predicted, the mazes do lead to different time courses, which fulfills property 7a (different time courses). In general, as the mazes increase in difficulty, the agent takes longer to complete (or give up on) them. When the agent does give up, though, it takes less time. This makes sense since the agent is stopping early. Still, the maze with multiple subtasks takes longer than the maze with a single subtask when the agent gives up. The impossible maze takes slightly less (but still statistically

significant) time to give up. This is because, after the first subtask, all subtasks are considered "bad" subtasks, whereas in the other mazes all subtasks are "good" subtasks. This should mean that there are more negative appraisals in the impossible maze, causing the agent to feel worse and thus give up sooner.

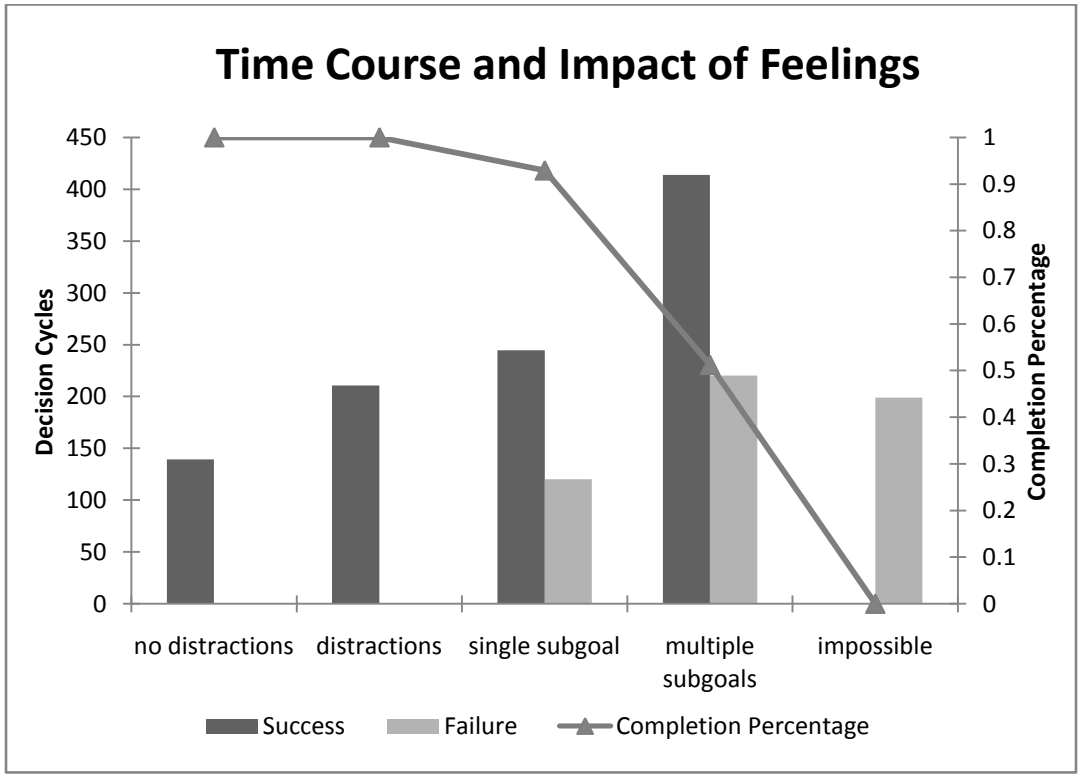


Figure 5.6: Number of decision cycles required to complete each maze. Success and failure cases shown separately. The line shows the success rate. All differences are statistically significant (1000 trials for each maze, >95% confidence level).

In Figure 5.7 we see that the data are consistent with this analysis. The feeling labels in the figure are generated as described in section 5.1.1. In each maze's feeling profile, the positive feeling (elation-joy) instances outweigh the negative feeling (anxiety-worry and displeasure-disgust) instances except for the impossible maze, where the negative feelings dominate. We can also see that each maze produces a different feeling profile, and that feeling profiles also differ between the success and failure cases. This supports property 7b (different feeling profiles). In contrast, the failure cases for mazes 3 and 4, the positive and negative feelings are nearly equal. This is to be expected given that the subtasks are "good," the agent positively appraises every move it makes (since it

thinks it is making progress). Thus, this offsets the negative feelings to some extent. However, each negative feeling in a subtask represents an opportunity to give up, and these more frequent opportunities lead to failure.

This, together with the data from Figure 5.6 supports properties 1 (emotional behavior) and 8 (choices influence feelings). That is, success and failure (both absolutely and in terms of rate) are defined by different feeling profiles, implying that feelings do influence behavior. Furthermore, even within the same maze the success and failure cases have different profiles, implying that the choices the agent makes in those mazes impacts feelings and behavior.

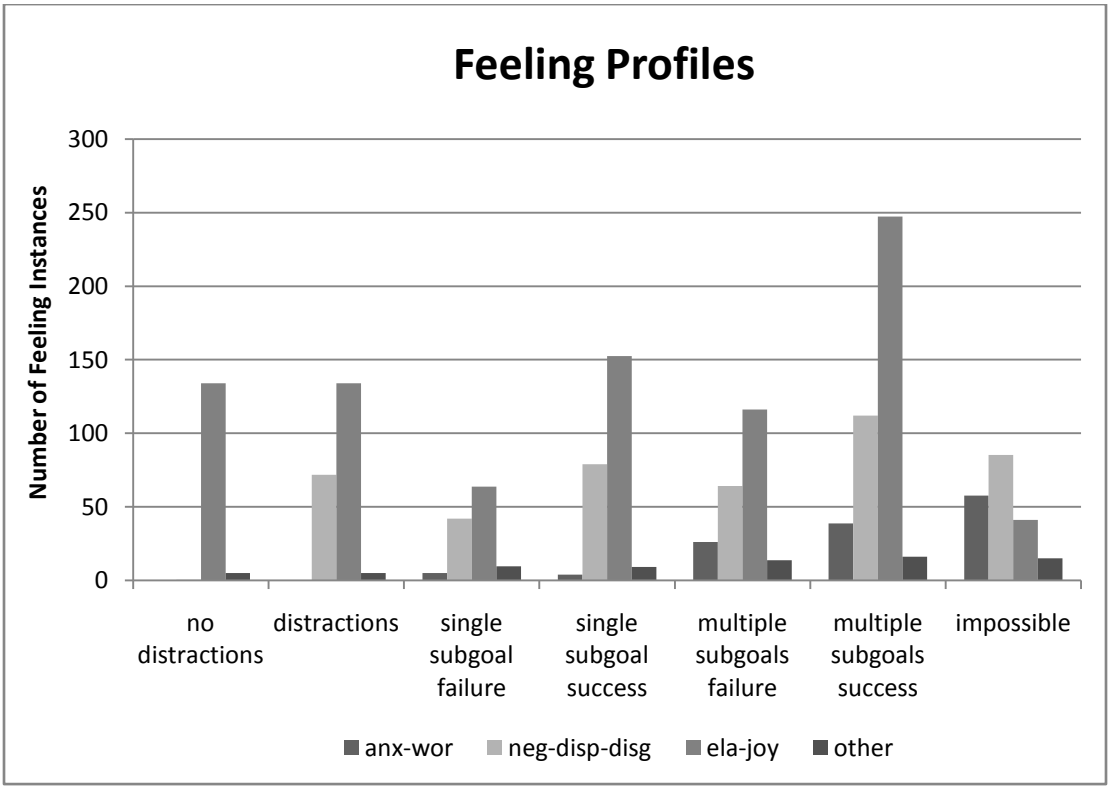


Figure 5.7: The average number of decision cycles each kind of feeling was active. Labels were produced by our labeling function. The success and failures for mazes 3 and 4 reported separately. "Other" includes Boredom-Indifference, Fear, Positive Displeasure-Disgust, and Sadness-Dejection. Differences between bars within a group (e.g., no distractions, etc.) are statistically significant (1000 trials for each maze, >95% confidence level).

Finally, the above analysis supports property 6 (purposeful, useful behavior). That is, the agent's behavior and feeling profiles are expected given its task and

environments. The agent completes the task in many cases, and when it fails, it has a negative feeling profile which justifies giving up.

As a final comment, as shown in Figure 5.7, the agent experiences a wide breadth of feeling types in these mazes (seven different kinds according to our labeling function). Given the limited nature of the domain, one might expect a much more limited set of feelings. Indeed, we have shown that multiple feelings can arise from simple manipulations of the environment, even in similar situations. One way is via interactions with the goal—adding structure that requires subtasks leads to many different feelings emerging. Another way is via interactions between mood (including decay) and emotion. Sometimes, even though we might classify a mood one way and an emotion another way, their combination results in yet another classification. This prediction could help explain why people are sometimes confused about their feelings.

5.3 An Intermediate Summary

Before moving on to our explorations into learning, we would like to summarize what we have learned so far. We have presented a novel integration of cognition and emotion based on the functional fit between appraisal theory and an abstract theory of cognitive control (PEACTIDM): cognition (as PEACTIDM) provides the processes necessary to generate emotions, whereas emotion (via appraisals) provides the data which cognition (via PEACTIDM) functionally demands. To evaluate the feasibility of this theory, we extended the Soar cognitive architecture to include the computational mechanisms necessary to support our proposed integration. We explored this system within the context of a simple stimulus response task and an ongoing task. Our evaluation centered on qualitative and quantitative issues regarding whether the system actually works and has features consistent with a complete emotion system. For the most part, it succeeds, although we discussed several avenues for future expansion.

We summarize the key theoretical features of our proposal as follows:

1. Appraisals are a functionally required part of cognitive processing; they cannot be replaced by some other emotion generation theory.

2. Appraisals provide a task-independent language for control knowledge, although their values can be determined by task-dependent knowledge. Emotion and mood, by virtue of being derived from appraisals, abstract summaries of the current and past states, respectively. Feeling, then, augments the current state representation with knowledge that combines the emotion and mood representations and can influence control.
3. The integration of appraisal and PEACTIDM implies a partial ordering of appraisal generation.
4. This partial ordering specifies a time course of appraisal generation, which leads to time courses for emotion, mood and feeling.
5. Emotion intensity is largely determined by expectations and consequences for the agent; thus, even seemingly mundane tasks can be emotional under the right circumstances.
6. In general, appraisals may require an arbitrary amount of inference to be generated. That is, the theory supports Marsella & Gratch's (in press) distinction between appraisal and inference.

This system lays the groundwork for extensive additional research. We present some of that research starting in Chapter 7, namely on intrinsically motivated reinforcement learning, and describe other areas in the future work at the end (Chapter 10). Next, however, we will briefly discuss the relationship of this theory in the context of related work.

Chapter 6

Related work

Like the Soar system we described in this paper, there are several implemented computational systems that use appraisal theory in some form and realize a functional agent that can behave in some environment, and in fact systems such as Gratch and Marsella's (2004) EMA (EMotion and Adaptation) inspired the current work. The primary goal of these systems is generating believable behavior, and there is less of an emphasis on the underlying theoretical integration of emotion and cognition, beyond the assertion that cognition is required to generate appraisals. In addition to different goals, these systems differ from the Soar system in two theoretically important ways. First, most existing systems generate appraisals and emotions all at once and then only rely on the emotion *outcome*. That is, while the emotion has an impact on the system, the appraisals do not. This property can be appreciated by observing that the emotion generation could occur via a non-appraisal process, and the system would not know the difference. In contrast, appraisal generation is required as part of the Soar agent's normal processing—they cannot be replaced by some other emotion-generation process.

Second, a consequence of appraisals being generated as part of the Soar agent's normal processing is that there is a time course to the generated appraisals and resulting emotions so that the during processing of a single stimulus, the agent's emotions can change as new information becomes available. Many existing systems do not support this because the appraisals are generated all at once.

In the remainder of this section we will briefly describe various systems with respect to these two distinguishing issues, as well as several other dimensions, including system type (architecture or modular), which appraisal theory is used, how many

emotions the system can have and whether they are categorical or continuous, and whether it has mood and feeling. Table 6.1 summarizes the comparison.

EMA is a computational model of a simple appraisal theory implemented in Soar 7 (an older version of Soar). EMA uses its own appraisal theory based on common dimensions from several existing theories. Like our model, appraisals are generated incrementally, but attention does not gate the generation of later appraisals. Rather, EMA generates multiple appraisal frames at once, and an attention mechanism focuses on a single frame, which determines the emotion. One or more categorical labels are then assigned to the single emotion instance; we interpret this as more specific emotion labels, as opposed to multiple emotions. EMA also has mood, which is an aggregate of all current appraisal frames; in contrast, mood in our system is an aggregate over previous emotions (including the current emotion). Finally, the appraisals are required by EMA's coping mechanism, but not directly by other mechanisms (e.g., the attention mechanism uses emotion intensity, but not the appraisals).

MAMID (Hudlicka, 2004) is a system aimed at building emotions into a cognitive architecture. MAMID's architectural mechanisms are higher level than Soar's, making it more a modular system by comparison. For example, it has a Situation Assessment module and an Action Selection module, as opposed building these out of more primitive components. Like EMA, the appraisals used are common to many theories. Unlike our system, MAMID generates an intensity for each of several categorical emotions. While this is modulated by the previous emotion, there is no separate mood concept. Appraisals in MAMID are generated "all at once," in the sense that the Affect Appraiser module takes in information about the current situation and outputs an emotional state. Thus, appraisal is not necessarily required by the system, and could be replaced by some other method for generating emotion.

Ortony, Clore and Collins (1988) describe a theory (commonly called the OCC model) that was not originally intended for use in systems that have emotion, but has since been implemented for that purpose. We will discuss OCC in the context of Neal Reilly's (1996) Em system. As a theory, OCC does not specify the architecture of the underlying system, but Em is implemented as a modular system. OCC uses a small set of

appraisals inspired by existing theories to generate an emotion hierarchy. In Em, multiple categorical emotions can exist simultaneously. OCC only briefly touches on mood, but leaves it unspecified. In Em, mood is an aggregation of current emotions, similar to how EMA uses an aggregate of current appraisal frames. Like MAMID, Em uses an Emotion Generation module that takes a situation description and outputs an emotion—the fact that it uses OCC (and hence appraisal) internally is not critical to its functioning. Like MAMID, then, appraisals are not generated incrementally.

Kismet (Breazeal, 2003) is a social robot. It is a modular system, but as a functioning robot, it handles real perception and motor. It also has physiological drives. While it has "appraisals," these are arousal, valence, and stance, which are better described as a circumplex model (Yik et al, 1999). Kismet can be in a single categorical emotion state at a time, and there is no mood (although the current emotion can indirectly influence the next emotion). Appraisal is not incremental, in the sense that all appraisal dimensions always have a value. Additionally, the appraisal information is only used to generate the emotions, and thus is not actually required by the system.

System	Appraisal Theory	Emotion Type	Mood/ Feeling	Incremental Appraisals	Appraisals required
EMA	Mixture	Categorical Single	Mood only	Yes	Coping only
MAMID	Mixture	Categorical Multiple	No	No	No
OCC/Em	Mixture	Categorical Multiple	Mood only	No	No
Kismet	Circumplex	Categorical Single	No	No	No
Our system	Scherer	Continuous Single	Yes	Yes	Yes

Table 6.1: Comparison summary.

Chapter 7

The Learning Model and an Initial Evaluation

The previous discussion begs the question: what functionality do emotions provide? As we alluded in Chapter 1,emotions may enhance aspects of the cognitive architecture's functioning. For example, memory retrieval may be enhanced by the use of emotion cues, or various parameters may be influenced by emotion. Perception and comprehension of a situation may be influenced by emotion. Physiologically, emotion may help prepare the body for action; e.g., fear might elevate heart rate and breathing, in case the agent needs to run away. Emotions are also widely recognized as important aspects of communication, via facial expression, tone of voice, and posture. Emotion may also play a role in learning.

In this thesis, our focus in on learning, although we do touch on parameter adjustment in Chapter 9. Our existing mood module could be considered a very abstract physiological model, but currently, detailed modeling of physiology is not supported by Soar (although we won't rule out the possibility for the future). We also have not explored emotion in social contexts yet, and thus have no model of emotional communication.

In this chapter, we present work in which reinforcement learning is driven by emotion. Intuitively, feelings serve as a reward signal that motivates the agent to learn to perform better (that is, to improve the feelings it experiences). The agent learns to behave in a way that makes it feel good while avoiding feeling bad. Coupled with a task that the agent wants to complete, the agent learns that completing the task makes it feel good. This work contributes not only to research on emotion in providing a functional computational grounding for feelings, but it also contributes to research in reinforcement

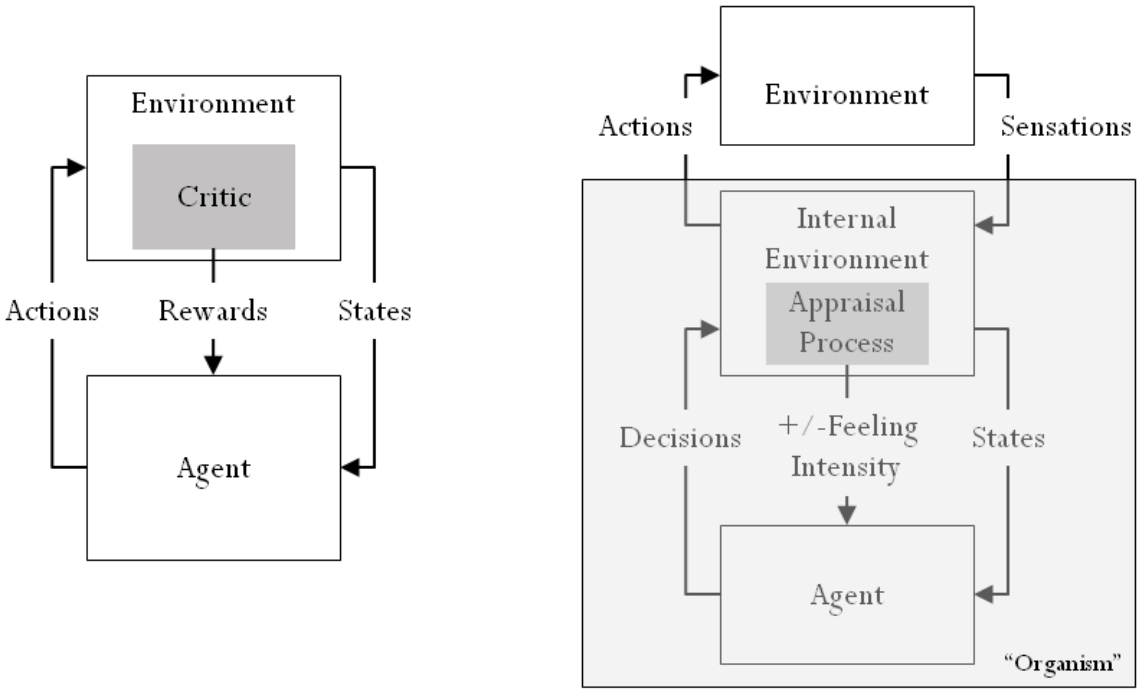
learning by providing a possible detailed theory of the origin and basis of intrinsically-motivated reward.

7.1 Intrinsically Motivated Reinforcement Learning

In traditional reinforcement learning, an agent perceives states in an environment and takes actions. A critic, located in the environment, provides a rewards and punishments in response to the choices being made (Figure 7.1a). The agent learns to maximize the reward signal (Sutton & Barto, 1998). This model is highly abstract and assumes a source of reward that is specific to every task.

In intrinsically motivated reinforcement learning, the environment is split into internal and external parts. The organism is composed of the internal environment together with the agent (Singh et al., 2004). The critic resides in the internal environment, and thus the organism generates its own rewards.

In our system, the appraisal process is the critic, and the resulting valenced feeling intensity provides the reward signal over which the agent learns (Figure 7.1b).



(a) **(b)**

Figure 7.1: Comparison of standard and intrinsically motivated reinforcement learning systems. (a) A traditional reinforcement learning system. The critic is part of the environment. (b) Our system viewed as an intrinsically motivated reinforcement learner. The critic is realized as the appraisal process, and is part of the organism. (Adapted from Singh et al., 2004.)

7.2 Related Work

As alluded above, Singh et al. (2004) implemented an intrinsically motivated reinforcement learning system. In their model, the agent generates intrinsic reward for unexpected events; this drives the agent to learn skills. That is, there is a set of actions the agent can take, and when they are combined together in the right sequences, the environment changes. For example, the agent can flip a switch, which may cause a light to turn on. In a more complex case, the agent can perform a complex series of actions (including things like turning on music, kicking a ball, and lower level eye and hand movements) that ultimately result in a toy monkey crying out. When these changes are new to the agent, it receives reward for causing them. As they become more familiar, reward diminishes. Thus, the agent is intrinsically motivated to learn skills. The system also incorporates extrinsic reward. The idea is that the agent can learn skills via intrinsic reward, and then utilize them in the pursuit of extrinsic rewards. For example, the agent

might get an extrinsic reward when the monkey cries out, but get intrinsic reward for all of the intermediate skills it must learn to do so (e.g., kicking the ball).

While this work demonstrates how intrinsically generated reward can lead to learning, it is very abstract and thus does not address any of the issues concerning integration with cognition or cognitive architecture. We could interpret the agent as generating an unexpectedness appraisal, but clearly this work is not in the context of a complete appraisal theory.

There have been other attempts to integrate emotion-like processes with reinforcement learning. Hogewoning et al. (2007) and Hogewoning (2007) describe a system developed in Soar that adjusts its exploration rate based on short- and long-term reward trajectories. They consider the reward histories to be a kind of affect representation. This work is differs from our own in that it emphasizes adjusting exploration rate, it is not based appraisal theories, and rewards are not intrinsically generated.

Salichs & Malfaz (2006) describe a system with the three emotions: happiness, sadness and fear. Happiness and sadness serve as positive and negative rewards, while fear affects the selection of "dangerous" actions. Happiness is generated when an external stimulus is present that is related to current internal drives (e.g., if hungry and food is present, the agent will be happy). Sadness is when the desired external stimulus is not present. Fear is when the state values have a large variance (even if positive overall). This work is interesting in that it connects physiology to goals and thus emotion. However, its range of emotions is limited, and there is no underlying theory that unifies the emotions (they are each determined separately), nor is there a principled integration with cognition.

7.3 Reinforcement Learning in Soar

As noted above, we generated a reward signal using the agent's current feeling. To take advantage of this reward signal, we used Soar's reinforcement learning mechanism (Nason & Laird 2005). This mechanism works by learning expected values for future reward to aid in the selection of operators. This knowledge is represented in Soar via numeric preference values associated with operator proposals. These numeric

values are created by rules called *RL rules* that match a proposed operator in a given context (as determined by the conditions of the rules). A given RL rule may be very specific or very abstract, and thus may match in many or few contexts. Thus, there is no single state representation; rather, the state is determined by the features tested by all of the conditions of the RL rules that match at any given time. When multiple operators are proposed in a given situation, the agent will select the one with the highest value (in an epsilon greedy fashion (Sutton & Barto 1998)). For example, if operator A is proposed with a numeric preference of 0.3, and operator B is proposed with a numeric value of 0.8, operator B will be selected unless an exploratory move is made, in which case the selection occurs uniformly randomly. The agent can receive a reward in the (which occurs when a rule places a numeric value in a special place in Short-Term Memory). In the Decision Phase, if there is a reward available, the values of the RL rules involved in the selection of the current operator are adjusted using a variant of the SARSA algorithm adapted for Soar (Nason & Laird 2005). This, in turn can change behavior as the RL rules associated with operators that lead to the most reward will tend towards higher values than other RL rules.

7.4 Reinforcement Learning in the Model

The reward signal was generated as follows: the magnitude was determined by the intensity of the feeling, and the sign was determined by the valence of the Conduciveness appraisal. Thus, the reward signal falls into the [-1,1] range.

In order to allow the agent to learn, we added several RL rules associated with various decision points. Thus, instead of making random decisions about, say, which direction to move in given a set of Encoded structures, the agent instead learns which actions to take as it gains experience, e.g., those that lead it closer to the goal.. Thus, the agent learns which stimuli to Attend to, whether it should ignore an Attended stimulus or Intend taking an action, when to create subtasks, and when to retrieve supertasks. Because of this, the potential for inferior performance exists: for example, the agent can ignore a stimulus that is on the path to the goal, or it can Intend moving into a wall. The agent will have to learn not to do these things. Table 7.1 summarizes what the agent can and cannot learn.

Perceive/Encode	No learning; for a given situation, the same structures are generated.
Attend vs. Tasking	The agent learns to choose among four or five stimuli to attend to and two subtasks to create (or one to retrieve). Thus, it must choose from between five to seven possible operators in this step.
Verify vs. Ignore	The agent learns when to process the stimulus further. However, once chosen, both routes have a fixed outcome (i.e., the agent does not learn "how" to do them).
Intend	No learning. Once the agent has Attended and verified a stimulus, the action is determined (move in the direction associated with the stimulus).

Table 7.1: Summary of what the agent can and cannot learn.

The features tested by the RL rules associated with each of the operators are summarized in Table 7.2. As we can see, it's actually fairly complicated; we discuss how we simplified this in the next chapter.

Feature	Operator
Passable	Attend, Tasking, Ignore, Intend
Path	Attend, Tasking, Ignore, Intend
Good/bad Subtask	Attend, Tasking, Ignore, Intend
X,Y location	Tasking
Features across multiple stimuli	Tasking
Subtask type	Give up

Table 7.2: Summary of features tested by RL rules associated with various operators. Tasking includes both Create Subtask and Retrieve Supertask.

The reader will notice that the features here are not the agent's current feelings. We did explore using the feeling frame values as features, but it did not work (results not shown). The issue is that most appraisals are continuous in nature; thus, in order to have RL rules that match in multiple situations, the features would need to be value ranges. Unfortunately, even with a small number of bins, this still lead to a large number of RL rules, each of which only matched in a small number of situations, which meant learning was slow. Additionally, from a theoretical standpoint, there's no reason to believe that the current values of the agent's feelings will tell it much about what it should do next, since it includes historical information which can skew the description away from the current situation. Thus, while the feelings were used to generate rewards, they were not used as the state features. The reader might wonder about using the emotion frame as

state features; however, as the emotion frame is derived directly from the above features, this would have been equivalent to using those features, which is what we did.

We also extended the description of a subtask to include a route to the subtask location. For example, if the subtask is to reach x=4, the description would specify whether that would be accomplished by moving north or south. This is to allow the agent to learn distinct values for each of these routes, since they may not both lead to the goal. The task is shown in Figure 7.2 with the optimal subtask locations marked.

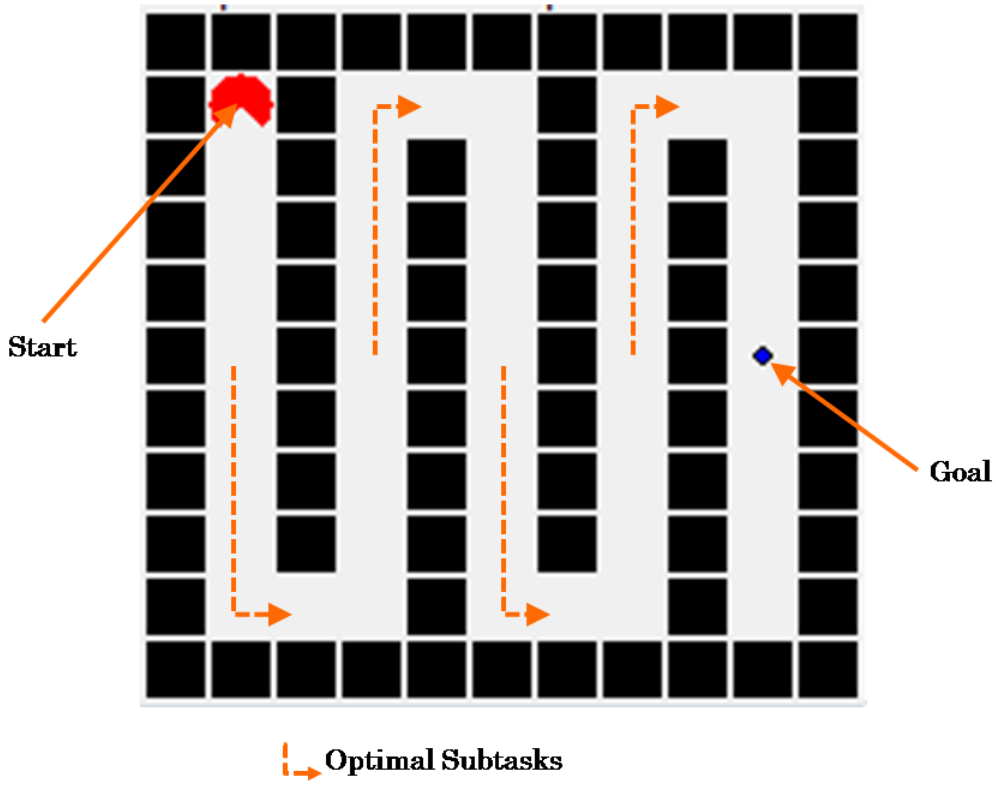


Figure 7.2: The maze used for the learning tests.
The paths of the optimal subtask locations are shown.

Additionally, minor changes were made to the exact values of several of the appraisals. These values are still qualitatively the same as we reported earlier but were tweaked to improve learning. For example, sometimes the agent would get stuck in cycles where it could get infinite reward; adjusting the values slightly was sufficient to

overcome this problem in this simple domain. We will return to the cycle problem again in Chapter 9 where we provide a better solution. We also disabled the agent's ability to give up. This allowed us to assess the ability of the agent to learn without interference from this additional effect. We could have explored learning when to give up, but decided not to in order to simplify the learning problem the agent faces.

While the problem may look very simple (there are no branches in the maze), from a reinforcement learning perspective, learning to solve these problems has its challenges. First, the problem is partially observable. The RL rules associated with most operator proposals do not include the x,y location of the agent (the create-subtask operator proposal is an exception[4]). Second, the agent always has four stimuli to contend with (five when the goal completion internal stimulus is present); this is exacerbated by the inability to determine what state it is in. Additionally, it has to contend with whether it should be Tasking or not, and how. Third, the problem is non-Markovian; that is, the available sensory information at each step is not guaranteed to be sufficient to uniquely determine the best action and additional historical information may be required. For example, the agent doesn't know what direction it just came from, and thus can't easily avoid backtracking. Mood provides some historical information, but also contributes to the non-Markovian nature of the problem. This is because mood influences the current feeling, which determines reward, and mood is influenced by previous states. Finally, the agent doesn't know the effects of its actions (e.g., it doesn't know that moving into a wall will get it nowhere). These factors make this task quite challenging.

Still, this task is far from impossible. Recall that each Encoded structure contains information about whether the associated direction is on the path towards the goal or not (section 4.1.1). Because this is generated by dynamic difference reduction, a wall might be marked as "on path" because, if the agent could move there, it would indeed be closer to the goal. Thus, the agent essentially has to learn to follow the stimuli that are on path, unless the only on path stimuli is not passable, in which case it needs to learn to create the proper subtask to get around the blockage (i.e., moving north vs. moving south).

[4] Providing the x,y coordinates for create-subtask operator proposals reduces the partial observability; this was simply to make the problem easier.

Still, if it simply moves past a point where it should be creating a subtask (see Figure 7.2), then moving backwards will actually be marked as on path and passable, and thus lead to short-term reward, even though it does not actually get the agent closer to the goal. Once a stimulus is Attended, it must learn whether to ignore it or not. Generally, the agent will eventually learn to Attend to the right stimulus, and thus needs to learn to not ignore it, but the agent may learn to ignore a "bad" stimulus that is Attended because of an exploratory move.

7.5 Methodology

Three agent types were tested: a standard reinforcement learning agent, which only received reward at the end when it accomplished the task, an agent that had no mood (so its feelings were its emotions) and a full agent that included mood.

We expect the standard reinforcement learning agent to have difficulty since it does not have access to the sensor that tells it how far it is from the goal. This may seem like an unfair comparison; however, creating a standard reinforcement learning agent with this capability but without the other appraisal information is difficult, since the appraisal representations comprise part of the state representation. If we were to remove the appraisal information, then the standard reinforcement learning agent would really be solving a different problem. If we leave the appraisal information, the agent is not really standard. However, the agent without mood can be viewed as a very rough approximation of an agent that would take advantage of this information to generate more frequent rewards. This approximation includes appraisal information, but without mood it is not the complete emotion system. Thus, we have two extremes and an intermediate agent: an agent with no emotion information at all, an agent with emotion but no mood, and an agent with both emotion and mood.

The agents learned across 15 episodes. This was repeated in 50 trials. We recorded the amount of time it took each agent to complete the task (measured in Soar decision cycles). Because of the task difficulty, the agents would sometimes get hopelessly lost; thus, to limit processing time, episodes were cut off at 10000 decision

cycles. We report the median to avoid skewing the data.

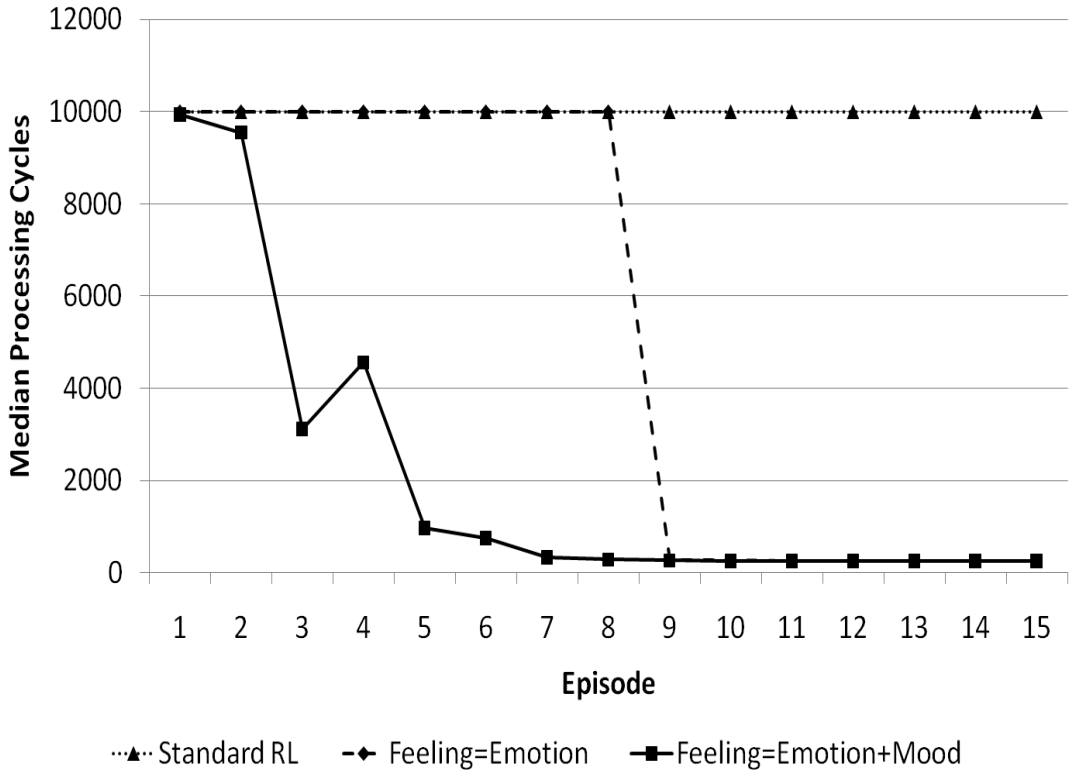


Figure 7.3: Learning results for three different agents.

7.6 Results

The results are shown in Figure 7.3 and Figure 7.4. The horizontal axis is the episode number, while the vertical axis is the median amount of time it took the agent to complete the task.

First consider Figure 7.3. The standard reinforcement learning agent never made any significant progress. This is expected because 15 training episodes do not provide the agent with enough experience when it only gets a single reward for each episode. In reinforcement learning, the reward "backs up" only one state per episode, and there are many more than 15 states on the path to the solution in this domain[5].

[5] Calculating the exact number of states is difficult due to the nature of RL in Soar. However, we estimate there to be more than 500 states.

The agent whose feeling is just its emotion (without mood) does not appear to be learning at first, but the values eventually converge. The agent whose feelings are composed of both emotion and mood does much better earlier on, learning much faster.

Figure 7.4 shows a close-up of the last several episodes for the two agents with emotions. The "error" bars show the first and third quartiles, which gives an indication of the amount of variability in the agents' behavior at that point. As we can see, the median for both agents reach optimality at the same time, but the variability of the agent with mood is much lower. In fact, the variability of the moodless agent reaches all the way to 10000 even in the final episode, implying that fewer agents did well on the task. In contrast, by the final episode, the agent with mood has small variance.

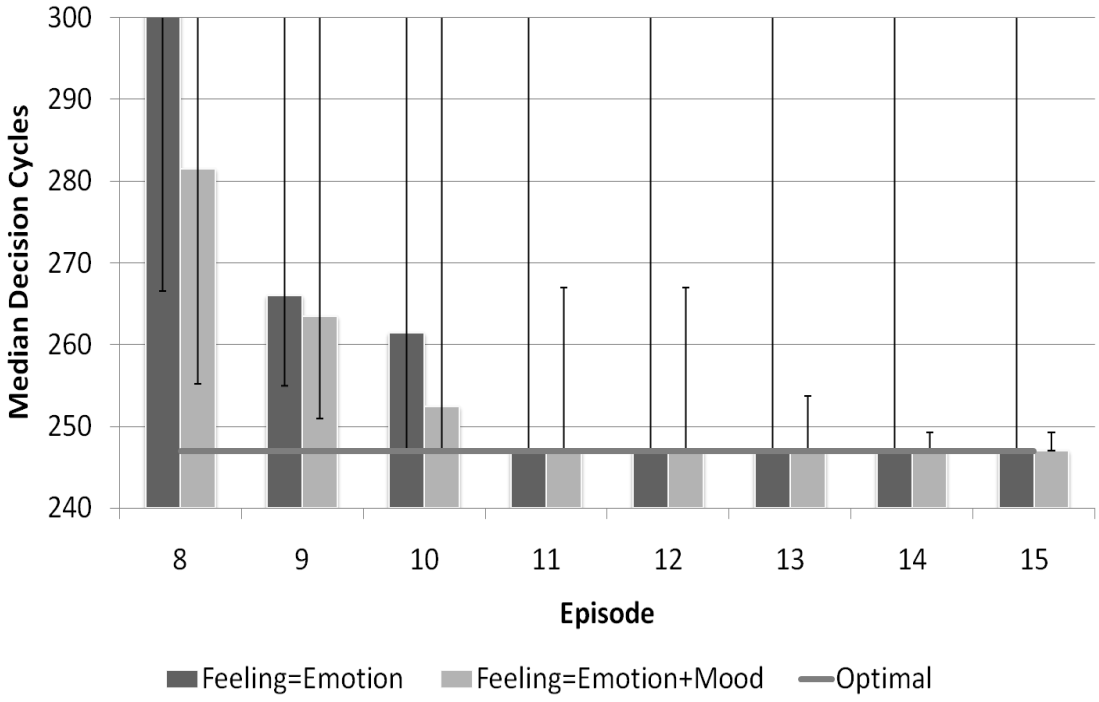


Figure 7.4: Close-up of last several episodes for agent with just emotion and agent with emotion and mood.
"Error" bars show first and third quartiles.

7.7 Discussion

The first thing to note is that the agents with emotion learn very fast relative to the standard reinforcement learning agent. This is because they get frequent reward signals (on every decision cycle) and thus get intermediate feedback on how they are doing. The standard reinforcement learning agent only gets reward feedback at the end, and thus it takes a long time for that information to propagate back to earlier actions. This result is not unexpected since part of the agent's feeling is related to whether it is getting closer to the goal or not (albeit with the caveats mentioned earlier, it was not clear that the agent would be able to learn at all).

Next, the agent with mood learns faster. The reason is because, sometimes when the agent is doing some internal bookkeeping kinds of processing, it is not experiencing an emotion. Thus, the agent without mood will get zero reward for those states, and later reward has to propagate back through those states. Propagation takes time (this is why the standard reinforcement learning agent takes so long to learn).

The agent with mood, however, carries a summary of its recent emotions forward into those states (with some decay). Thus, these states get reasonable value estimates, which speeds up the propagation immensely.

The reader may be concerned that all we have shown is that intermediate rewards speed learning, regardless of their origin. We agree that meaningful intermediate rewards should always speed learning; what we have presented is a theory about the origin of those rewards and how they are applied to an integrated system centered on abstract functional operations, and a demonstration that the rewards generated by that theory do, in fact, speed learning.

Chapter 8

Revising the Model: Extending to a Continuous Domain; Improving Simplicity and Correctness

8.1 Open Questions

Based on our experience with the previously described model, many questions arose, including:

- Will our model work in a more complex domain? That is, will it scale to a continuous time, continuous space domain?
- Which aspects of the model are responsible for its performance? That is, which appraisals really make a difference in learning?
- Is the model more complicated than it needs to be? That is, can some aspects of the agent be unified or eliminated to produce a simpler model that allows us to more directly answer the previous question?

In order to address these questions, we created a new domain and streamlined the model. Also, rather than use the full model for the task, we chose a subset of appraisals to develop and analyze more fully. In this chapter, we will describe the new domain and the revised model.

8.2 A New Environment and Task

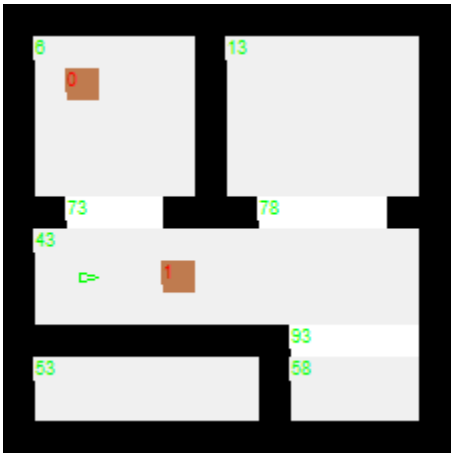


Figure 8.1: The room environment.
The agent is the green triangle; starting configuration for this map shown. The numbers show the ids of the rooms and blocks (the ids of gateways are not shown). This map has seven rooms the agent can see (the lower-left room is required to make the map rectangular, but is not part of the agent's environment), six gateways, and two blocks. The storage room is id 13 (upper-right).

The new environment is called the Room environment (Figure 8.1). It is a 2D world, with rectangular rooms connected by doors (1-dimensional gateways). Rooms may contain zero or more blocks. The environment is discrete in the sense that walls, blocks, etc. are located on an underlying grid (for example, a block takes up one grid space), but continuous in that the agent can move in a continuous fashion (that is, unlike Eaters, the agent does not move from grid location to grid location discretely). Each object (room, gateway, or block) has a unique id, which the agent can perceive. The agent can also perceive the locations of the objects in terms of their angle and distance relative to itself. Areas and gateways are always visible to the agent, while blocks can only be seen if they are in the agent's vision cone. Time in this domain is discretized at the decision cycle level (so time moves forward one step for each decision cycle in Soar, regardless of whether the agent is doing anything). The actions the agent can take include turning and moving forward (at fixed rates each time step), picking up a block (if in an adjacent grid space) and putting down a block (in the grid space in front of the agent). The agent can move forward and turn at the same time, and can only carry one block at a time.

The task we created for the agent in this environment is called the clean house task. The world represents a house with some blocks scattered throughout. One of the rooms is designated the storage room. The agent's task is to clean all the rooms. This involves moving any blocks it finds to the storage room. Even if a room has no blocks, the agent must still visit it in order to know that. Furthermore, the agent does not know the layout of the house, so it must find out how many rooms and blocks there are. Thus, although the agent knows the id of the storage room at the beginning, it must still find that room.

Does this environment and task fulfill our needs? The room environment is certainly more complex than the Eaters environment, including continuous time (essentially) and space, and more actions. The task is also more complex, involving moving objects around as opposed to just moving the agent around. Thus, this should give us ample opportunity to see if the model scales.

In terms of allowing us to test the influence of various appraisals, the domain should provide enough features to develop models for several appraisals, but not all (for example, as a single-agent task, there is not much opportunity to explore causality). Still, it is sufficient for exploring the premise of the thesis; namely, to see if this kind of a system can work at all (or, having accomplished that, to explore why it works). Finally, this domain certainly doesn't impede us from trying to simplify the model.

8.3 Revising the Model

There were multiple issues with the previous model that we addressed via various revisions:

- The model requires different domain-specific knowledge (8.3.1) (e.g., how to encode the stimuli for this domain and what the subtasks are). This wasn't an issue with the previous model; it was just a basic requirement. The model was also extended to include the ability for the agent to create a task-specific map of the environment; this was necessary primarily because, without knowledge of where the blocks are and what rooms are clean, the agent would have no way to know when it had completed the task.

- The previous model had issues with prediction that resulted from the way Tasking interacted with Intend. This was addressed by unifying Tasking with Encoding and Intending (8.3.2).
- Previously, the theory determined the valence of the reward based solely on the Conduciveness dimension. As we looked more closely at other appraisals, it seems that more than just Conduciveness should contribute to valence. This led us to change not only the calculation of valence, but also the calculation of the resulting reward (8.3.3).
- There were some changes prompted by the continuous nature of the environment (8.3.4). Specifically, we adjusted the time appraisals continue to influence emotion, mood and feeling, and how reward should back up between temporally distant states.
- Some aspects of the model complicated learning and obscured the core issues (i.e., ignoring stimuli and giving up), or introduced arbitrary constraints (i.e., inability to Attend to the same stimulus twice). These aspects were removed to simplify the model (8.3.5).
- There was an error in the way feeling intensity was calculated that was corrected (8.3.5).

We will address each of these in turn.

8.3.1 Domain-specific knowledge

There are three kinds of external stimuli, one for each of the objects in the world: rooms, gateways, and blocks. As in Eaters, there are also internally generated stimuli for completed tasks. The agent has one top-level (super) task, which is to clean the house, and two subtasks: clean the current room, and go to another area. In this domain, each task type has a unique completion stimulus associated with it.

The agent creates an Encoded structure for each stimulus containing information used to drive behavior directly and to support the creation of appraisals. A summary of the Encoded information is shown in Table 8.1; details are discussed below.

Common	Id, Type, Passable, Path, Distance-to-goal, Progress
Room	(Common)
Gateway	To-room-id, Range, Angle
Block	Range, Angle
Task-completed	(Common)

Table 8.1: Encoded features for various stimuli.

For direct behavior, each stimulus contains information directly from perception such as its type and id; these are used by the agent to determine what kinds of actions can be taken in response to a stimulus, and to distinguish stimuli of the same type. Gateways and blocks also contain the angle the agent must turn to face it and the range (distance) to the object, which is used to determine how to physically move to it. Gateways include the id of the room to which they connect, which is necessary for the agent to determine if parts of the world are not yet explored..

For appraisal generation, the Encoded information is further augmented with internal information including whether the stimulus is "on the path" to the current task, the distance to the goal, and whether the agent is making progress. The path and progress augmentations help determine the agent's Goal Relevance and Conduciveness appraisal values. This is described in the next chapter.

Each stimulus is labeled as "on the path" or "off the path". For example, if the agent is already carrying a block, then attending to another block would be considered off the path. On the other hand, if the agent's current task is to go to some room, and there is a gateway that leads to that room, the corresponding stimulus would be on the path. Task completion stimuli are always on the path. Path information comprises a lot of knowledge that helps guide the agent; we will revisit this later.

Distance to the goal (not to be confused with range to objects) is calculated in the following way. For the clean house task, the distance is the number of rooms not yet cleaned, plus one if there are any unvisited rooms (since there may be entirely unknown rooms in that case). For the clean room task, the distance is the number of known blocks currently in the room, plus one if the room needs to be checked (recall that the agent can only see blocks in its vision cone, so there may be blocks it is unaware of in the room).

For the go to room task, the distance is the number of rooms the agent must traverse to get to the destination room from its current location.

The purpose of computing distances is to support the generation of the progress augmentation. An agent is making progress if the distance to the goal is less on the next attend operator. In the case where the task has just changed (because the agent created a subtask or retrieved the supertask), progress is determined by whether the agent was making progress previously (that is, when no previous distance is available to compare to, we assume the agent is doing as well as it was).

With respect to intentions, each stimulus can only be responded to in a single way, based on context. This is to focus the learning problem as discussed in the next chapter. Thus, if attending to a gateway, the agent will move through that gateway. If attending to a block, the agent will move to the block and pick it up (unless it is already carrying a block, in which case it will just move to the block and do nothing). If attending to a room other than the storage room, the agent will spin in place. This allows it to see all the blocks in the room (or confirm that there are no blocks in the room). If attending to the storage room, the agent will put down the block it is carrying. If it is not carrying a block, then it will just spin.

We don't consider this limitation particularly restrictive. In the Eaters model, the agent already had the choice to give up or ignore instead of Intending; adding additional possible actions would be similar.

Finally, the model was extended by giving the agent the ability to create a task-specific map. The agent creates a map of the world as it moves about; thus, it knows which rooms are connected to which, whether they have been visited, and if it has seen any blocks in them.

This map is built using operators that are not related to appraisals. In the PEACTIDM framework they could be considered Comprehend operators, but their use was primarily pragmatic, and not a serious attempt to model how people would do this. The information in this map is used to determine when the clean house task is finished (all rooms in the map are marked as clean). It is also used to calculate the distance for the

go to room subtask. Finally, it is used to calculate some of the path information (e.g., if not carrying a block and the clean house task is active, then creating a subtask to go to a room that contains a block is on the path).

8.3.2 The Unification of Tasking with Encoding and Intending

In the previous model, Tasking was handled by different operators than Attending. That is, the agent could choose to Attend to a stimulus or to engage in Tasking. This resulted in some additional complexity: if the agent chose to Attend, then it would follow a Comprehend-Intend path and deal with verifying predictions, whereas if it chose to Task (e.g., create a subtask), then it has to use a different set of operators that did not involve Comprehension (including verifying predictions) or Intending.

This caused problems with predictions. First, the agent's predictions were generated in the context of the current (sub)task, which meant that if the agent switched tasks, the prediction was invalidated. This would lead to many cases where there was no prediction. Furthermore, this made the prediction aspects of the system fragile (predictions had to be dealt with properly in all possible edge cases where they might be invalidated, and the system had to know what to do when there was no prediction). The root of the issue is that Tasking is essentially an internal action, but it doesn't generate or process predictions in the same way that external (Intended) actions do.

Furthermore, this separation made the learning problem more complex (at least conceptually), because the agent had to separately learn not only what to Attend to, but whether it should be Tasking instead. Maintaining this distinction was challenging and unnecessary.

In the revised model, Tasking opportunities (i.e., creating new subtasks or retrieving the supertask) are all Encoded as internal stimuli (Figure 8.2), similar to how task completion stimuli are Encoded internally. As with all stimuli, the agent always makes predictions about what stimulus will be perceived next; thus, the agent can predict that it will perform Tasking next, or what stimulus it will Attend to following a Tasking operation. The Tasking operation itself is performed in the Intend step. Thus, the agent's cycle is always "Attend-Comprehend-Intend." Does this imply a major change to

PEACTIDM? Not really; it is essentially a matter of perspective and semantics. We could either say we have changed PEACTIDM by integrating Tasking with Encode, Attend, and Intend, but we could just as easily say that, from the PEACTIDM perspective, these are completely separate processes, and the fact that they share operators is an implementation detail. Still, it is an important implementation detail that has simplified the code, but also unifies the theory in an interesting way. Thus, we prefer to think of it as a theoretical modification.

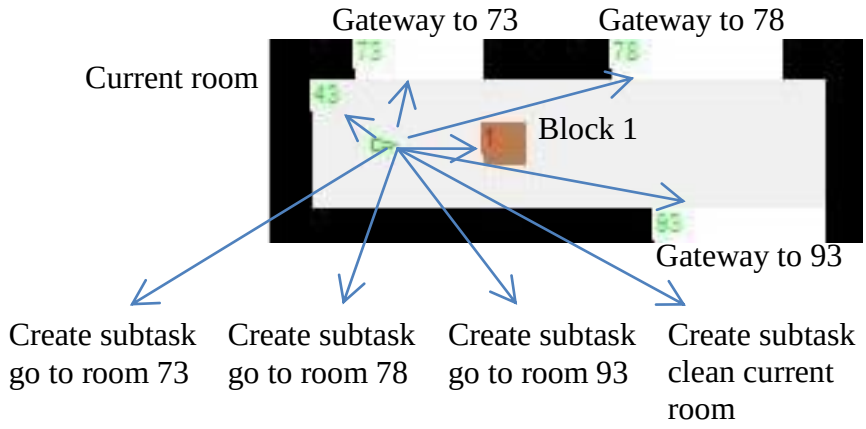


Figure 8.2: Example of stimuli the agent must choose among for Attention. This example is when the agent has just started (and thus only knows about rooms immediately adjacent to the starting room). As the agent learns about more rooms, it will have stimuli to create subtasks to go to those rooms, too.

8.3.3 Valence and the Calculation of Reward

In the Eaters model, the magnitude of the reward was the feeling intensity, and its valence was the valence of the conduciveness dimension. The problem with this approach is that more dimensions than just conduciveness appear to contain valence information. In particular, those dimensions whose values can be negative imply a valence. For example, intrinsic pleasantness can have a value that implies intrinsically unpleasant, which should impact the valence.

To accommodate this, the valence needed to be represented as a continuous value instead of just a sign. That way, different appraisals with valences can interact to produce an overall valence. This is accomplished merely by averaging the valences

together. Averaging means the final valence value will be in the [-1, 1] range, just like the individual appraisals that contribute to it.

But how do valence and intensity interact, then, to produce a reward? For inspiration, we turn to circumplex models of emotion (see Yik et al., 1999 for a review). In circumplex models of emotion, there are commonly two dimensions used to describe an emotion: intensity (variously called arousal, activation, engagement, etc.) and valence (also called pleasantness). Various combinations of these dimensions are associated with various emotions (Figure 8.3). Our interpretation of circumplex theory is that, unlike appraisals, which describe emotion antecedents, these dimensions describe what an emotion looks like when it is manifest.

With regards to reward, we consider some cases. Suppose intensity is very high (near 1), but valence is very low (near 0). In this case, the reward should be a medium value, since the stimulus is simultaneously important (intense) but unclear (not strongly valenced). The agent, thus, should not be influenced too strongly by this outcome. Consider the case where the intensity is very high but the valence is exactly zero. This time, the reward should be zero because the stimulus is not punishing or rewarding—it is merely exciting (in a completely neutral way). A simple way of combining intensity and valence to produce these results is simply to multiply them. Thus, the reward still falls in the [-1, 1] range.

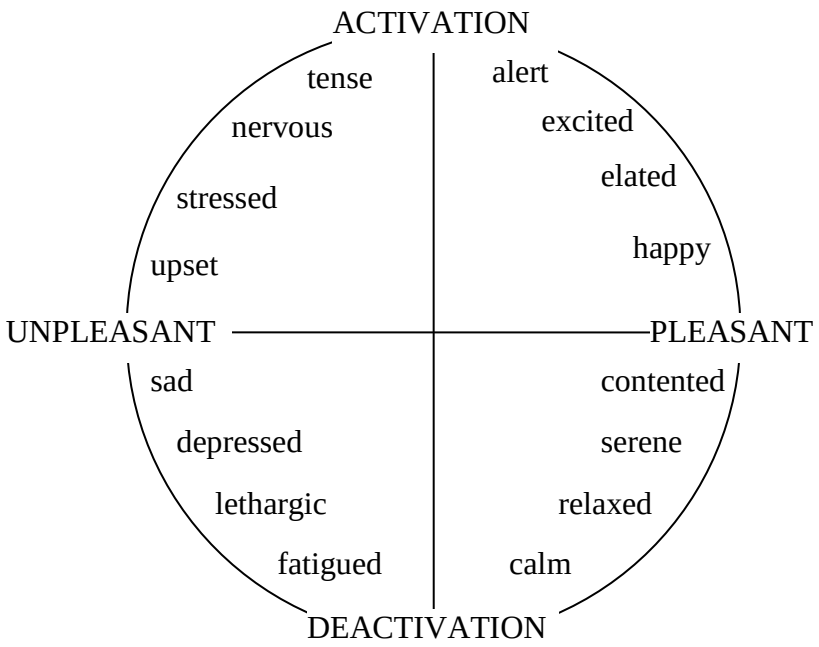


Figure 8.3: Typical circumplex model.
Adapted from Feldman Barrett & Russell (1998).

8.3.4 Adapting to a Continuous environment

The change to a continuous environment raised some issues related to the effects of time. First is the issue of the duration of an emotion. Previously, an emotion lasted the entire time the corresponding appraisal frame was active. In the discrete domain of Eaters, this meant a few decision cycles before the agent was ready to attend to another stimulus. In the continuous time and space of the room environment, the agent is attending to a stimulus for the entire time it takes to execute the resulting intention. For example, if the agent is attending to a gateway on the other side of the room, the appraisal frame for that stimulus will be active as the agent moves across the room, which may be over 100 decision cycles. This can cause major skewing of the reward: actions that take longer will be rewarded or punished more. Mood will be similarly skewed. To prevent this, an appraisal frame is deactivated as soon as the intention begins (in Soar terms, when the Intend operator is selected). Thus, the appraisal frame is only active for two decision cycles: Attend and Comprehend. It is during this time that the agent may receive non-zero reward; the rest of the time reward is zero.

Second, there is the issue of how reward should backup to previous states that are separated by time. In the discrete case, every decision cycle corresponded to a reinforcement learning state, even if it was a fixed decision (that is, every operator had an associated RL rule). This resulted in many RL rules whose only purpose was to allow values to backup from later operators. Furthermore, the exact descriptions of these states could influence how values backed up through them. To circumvent this issue entirely, Soar's reinforcement learning mechanism was given the ability to "jump gaps" between RL states. Thus, two RL states can be separated by as many decision cycles (that is, time) as necessary and reward will backup from one to the other (discounted by the number of decision cycles in between). For example, suppose the agent Attends to a gateway across the room. The next stimulus it Attends to might be the new area it ends up in. With gap jumping, the agent will not need an RL rule for every decision cycle in between (e.g., while it walks across the room).

8.3.5 Simplifications

The agent's ability to ignore stimuli was removed. This was so we could better focus the learning problem (see Chapter 9 for more details). This has behavioral implications as well—if the agent Attends to a stimulus, it commits to performing the corresponding action, which may be lengthy (e.g., walking across the room). Without the ability to ignore, the agent is stuck doing this action (even if it was chosen as an exploratory move). This is actually not out of line with how ignore was originally intended to be used; in the non-learning Eaters, it allowed the agent to avoid acting on impassable stimuli (i.e., walls). There are no such stimuli in the Room domain, so leaving out ignore does not represent a major change in capability.

8.3.6 Corrections

Another change has to do with how feeling intensity is calculated. Previously, dimensions whose values are in the [-1,1] range were "normalized" by dividing by 2. Ostensibly, this was to put them in the same range as those dimensions whose values are in the [0,1] range. Upon reflection, however, we decided this was a mistake. In the intensity calculation, we take the absolute values of these larger ranges, which already forces them into the [0,1] range. By dividing by 2, we essentially forced them into the

[0,0.5] range, which meant that they had less influence than the unnormalized dimensions. This "normalization" was removed.

8.4 Summary

The model was revised to work in a new continuous domain. Tasking was unified with Encoding and Intend. The way reward is generated was modified to allow for multiple sources of valence. Minor simplifications and corrections were made.

In the next chapter, we will explore the effects of various combinations of a subset of appraisals on learning in an attempt to determine how aspects of the system actually contribute to its performance. We will also explore how feelings can influence other aspects of the architecture.

Chapter 9

Learning Experiments and Evaluation in the Continuous Domain

9.1 What is the Agent Learning?

In the model, there are at least two kinds of things the agent can learn about: which stimulus to Attend to, and what to Intend in response to the chosen stimulus. (In principle, there are other things the agent could be learning, such as how to Encode stimuli, how decompose the task into subtasks, RL state representations, etc., but these things are not handled by our current theory or model.) In the revised model, we chose to focus on the Attention problem; thus, every stimulus in a given context has only one response (as described earlier). Given that Tasking options are now represented as stimuli, learning what to Attend to includes learning when and what to Task.

In Soar terms, this means there are only RL rules associated with the Attend operator. Those rules represent the state using the following features:

- task type (clean-house, clean-room, goto-room)
- task object (id of the storage room, room to clean, or room to go to)
- whether the task has been completed or not
- the unique id associated with the stimulus (e.g., the id of a block)
- path (on or off)
- progress (true or false)
- whether agent is carrying a block or not
- passable (always true in this task)

This representation is a further expansion of the representation used in Eaters. While features like passable, path, progress and current task[6] have been retained, new domain-specific features have also been added, such as whether the agent is carrying a block. Features such as the task object and the unique id of the stimulus are intended to be fairly domain-general, but may not be appropriate for some domains.

9.2 Choosing Appraisals to Explore

One of the goals of the revised model is to allow us to identify how various appraisals influence the agent's learning via their impact on the reward signal, and possibly the Q-values directly. In general, many of the appraisals in the previous model had very simple (even constant) calculations for their values. Thus, they probably exerted little influence on the outcome. Developing a theory for computing each appraisal could be the topics of several theses, and the value of trivial implementations is questionable. Thus, we opted not to include many of the appraisals. Furthermore, given our focus on learning what to Attend to (including Tasking), but not on how to Intend, some appraisals are more appropriate for this exploration than others. We explored a subset of the dimensions from the previous model: Conduciveness, Outcome Probability, Discrepancy from Expectation, Goal Relevance, and Intrinsic Pleasantness. We will discuss here which appraisals we included or not and why, and then discuss the appraisals we explored in more depth in the next sections.

First consider the Relevance appraisals (see Table 2.2). These appraisals are supposed to help the agent determine what to Attend to; unfortunately, most of them are too low-level in nature to be suitable for serious modeling in Soar. Suddenness is perceptually based; given that our implementation of the environment and task does not include a strong model of perception, we opted to leave it out. Similarly, taking Unpredictability seriously seems to require tracking low-level statistical data, for which the current model in Soar is not well suited. Goal Relevance, on the other hand, is a higher-level construct that we could easily model (see section 9.7). Our model also made it easy to explore Intrinsic Pleasantness (see section 9.8).

[6] In Eaters, progress and task were combined into a single good/bad subtask feature. These are now represented separately.

The next grouping in Table 2.2 is the implication appraisals. According to Scherer's theory, many of these dimensions, like Causal Agent, Causal Motive, Control and Power, are more directly relevant to choosing a response (Intending) than to choosing what to Attend to. Furthermore, for the room environment and clean house task, there is only one agent, making causality less interesting. Thus, these dimensions were also left out. However, the Conduciveness dimension is directly relevant to the model's learning (see section 9.5). Additionally, since the model already generates predictions, including Outcome Probability and Discrepancy from Expectation was natural (see section 9.6).

Other dimensions in Scherer's theory not mentioned above (Unfamiliarity, Urgency, Adjustment, and Internal and External Standards) were not included in the original model. Again, given that each one could be a thesis by itself, we chose not to engage in trivial explorations of them.

In the sections ahead, we will describe the models of the appraisals we chose. While these models are an improvement over the earlier models, they are still fairly simple. The goal here is to show that non-trivial models of these appraisals can contribute to learning in a complex domain. As stated above, a "complete" model of any particular dimension is well beyond the scope of this (and probably any single) thesis.

9.3 Methodology and Results Interpretation

The results below are of the agent solving the task in the environment shown in Figure 8.1 (storage room in the upper-right; see section 8.2 for details). Except where noted, the experiments were conducted with the following parameters: 50 trials of 15 episodes each. Learning rate and exploration rate held constant at 0.3 and 0.1, respectively; the exploration method is epsilon greedy. Because the agent can get stuck in cycles, episodes were cutoff after 10000 decision cycles. To avoid these outliers from skewing the data, medians are reported instead of means. Finally, to give an idea of the variability in the results, the first and third quartiles are also reported in the charts of decision cycles, total reward, and fraction correct predictions.

In interpreting the results, it should be noted that the first quartile of, say, the decision cycles chart does not necessarily correspond to the first quartile of the total

rewards chart. In fact, it is often the case that the third quartile of the decision cycles corresponds to the first quartile of the total rewards, because longer task times typically correspond to less reward. The only way to be sure (indeed, the only way in which it matters) is when a quartile has an unusual shape. In these cases, the unusual shape is often visible in the different charts, making it easy to see what corresponds to what.

Another result we report is failures. The failure data show the number of times the agent had to be cutoff (reached the decision cycle limit), which is usually indicative of a cycle (sometimes, but not necessarily, an infinite reward cycle) or just failure to learn. The failure figures show the total number of failures across all trials for each episode. These data can show trends in the way the agent fails as it learns. The failure tables show three kinds of failure counts. Total Failures is the number of episodes across all trials that were cutoff (total number of episodes is 50*15 = 750 for most experiments). Trial Failures is the number of trials that had at least one failed episode. Final Episode Failures is the number of trials whose final episode was a failure. We consider this number to be the most informative. If it is lower than the Trial Failures, it implies that the agent was able to recover from other failures (and if the cutoff was higher, that the other failure numbers would come down). If both Final Episode Failures and Trial Failures were high, then it would imply that the agent was learning cyclical behavior, or failing to learn at all. If both numbers are low, then it implies that the agent is learning well all the time. The failure figures and tables give complementary views of the failure data.

9.4 Overview of Results

In this section, we provide an overview of the results to help the reader.

1. Conduciveness (9.5): The agent learns to do well, but has a several failures.
2. Outcome Probability and Discrepancy from Expectation (9.6): The agent learns just as well, and failures are reduced to zero.
3. Goal Relevance (9.7): The agent's knowledge about important stimuli dominates performance, leading to very little learning (the agent does well right away). A reduced-knowledge version shows that the goal relevance improves performance,

but the other appraisals are still necessary for learning in knowledge-poor situations.

4. Intrinsic Pleasantness (9.8): This appraisal had a mixed influence on performance, which is expected for an appraisal that is independent of the goal.
5. Other results (9.9): Mood (9.9.1) had no effect. We also tried dynamically changing the exploration rate (9.9.2) and learning rate (9.9.3) parameters of the RL system based on the agent's feelings. That is, the feelings were used to regulate other parts of the cognitive architecture. This had a generally positive impact (faster convergence), but also resulted in slightly more failures.

A more detailed summary of the results is given at the end of the chapter.

9.5 Exploring Conduciveness

Perhaps the single most important appraisal, conduciveness, is the agent's most direct measure of how good or bad the current situation is for it. To review, the appraisals influence the feeling intensity and valence (indeed, if Conduciveness is the only appraisal, it determines them), which in turn determines reward. Thus, this appraisal alone is sufficient to generate a reward signal, and thus we expect the agent to learn to accomplish the task. That is, using this appraisal alone will tell us if the task is structured properly (with respect to how it is encoded, etc.) so that it is learnable at all. However, we expect other appraisals to help refine the reward signal or otherwise influence the agent's behavior in ways that result in improved learning (e.g., faster, or fewer errors, etc.).

9.5.1 Calculating Conduciveness

The calculation of Conduciveness was greatly simplified from the original model in order to make it easier to see how the system works (and to remove any mystery about what is really responsible for the results). In the revised model, Conduciveness is based only on the path (on/off) and progress (true/false) aspects of the encoded stimuli as shown in Table 9.1.

	Path: On	Path Off
Progress: True	1.0	-0.5
Progress: False	-0.5	-1.0

Table 9.1: Conduciveness values.

The values for "on path and progress true" and "off path and progress false" are self explanatory: it is good for the agent to be doing well and bad for it not to be doing well. The other values were chosen to be negative to avoid infinite reward cycles. An infinite reward cycle is a sequence of actions that leads to infinite reward, such that the agent learns to do that cycle instead of the task. Traditional RL domains do not have this problem because the agent only gets rewarded upon completion of the task (and rewards elsewhere are zero or negative). In a system where the agent generates its own rewards all the time, we must be very careful to structure the task such that the agent cannot get infinite reward by repeating the same set of actions over and over. For example, a common cycle that this value structure eliminates is creating a subtask that is on path, and then retrieving the supertask (which is off path), and then creating the on path subtask again, etc. If an on path stimulus were rewarding independent of progress, this could lead to an infinite reward cycle. Clearly, there are many possible value structures that would work, but we found this one to be adequate (it is still not perfect, but reward cycles are rare, so the agent does not usually find them).

9.5.2 Conduciveness Results

The experimental setup for this experiment is exactly as described above.

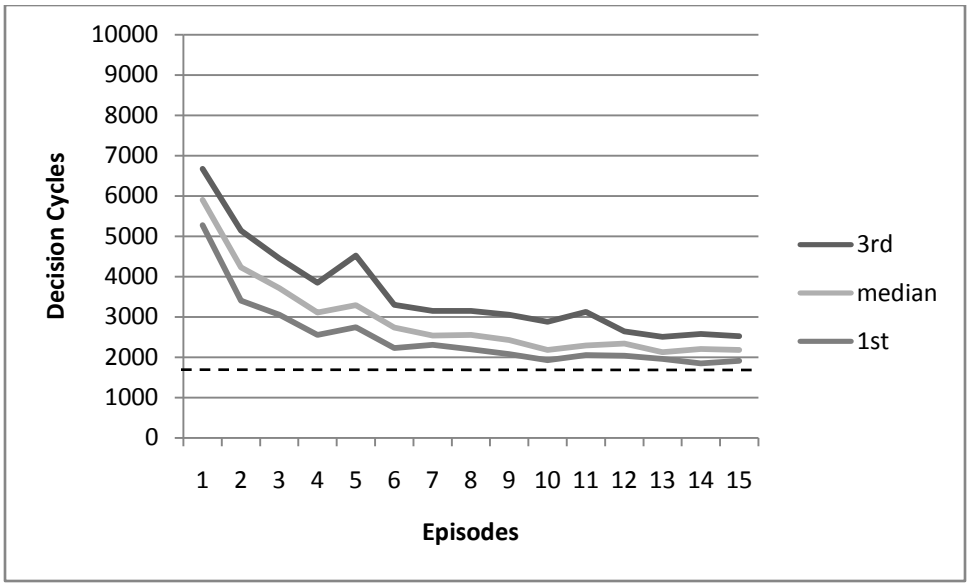


Figure 9.1: Decision cycles to task completion for Conduciveness experiment. The agent learns to complete the task in fewer decision cycles across several episodes. Median, first, and third quartiles shown.

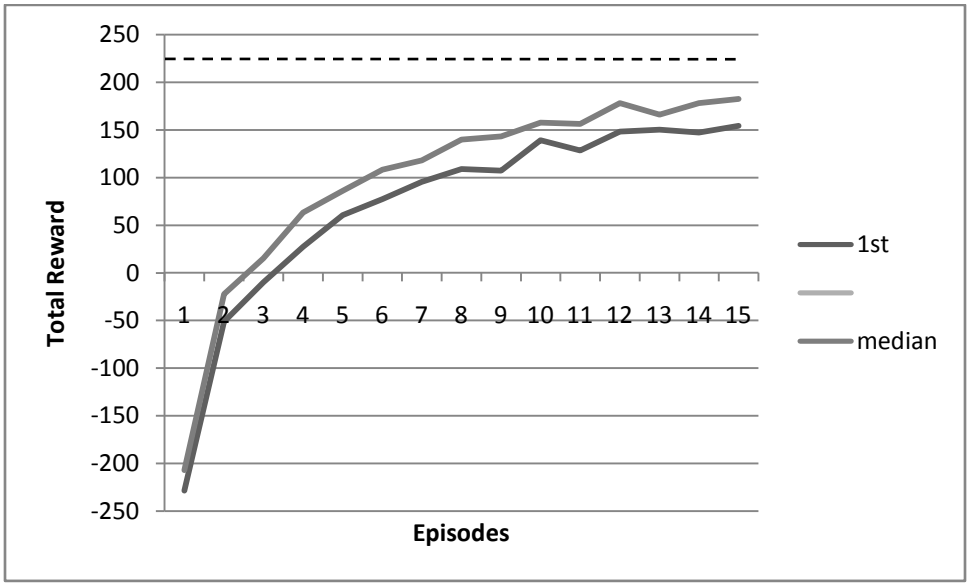


Figure 9.2: Total reward accumulated during task for Conduciveness experiment. The agent learns to accumulate more total reward across several episodes.

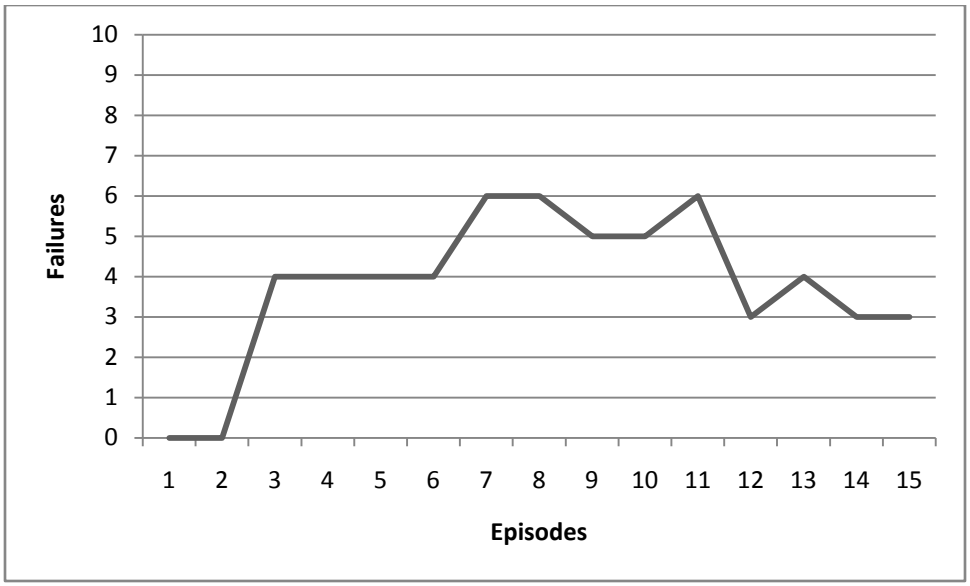


Figure 9.3: Failures across trials for each episode for Conduciveness experiment. Failures generally increase until the end where there is a reduction.

Final Episode Failures	Trial Failures	Total Failures
3	12	57

Table 9.2: Failures for Conduciveness experiment. The agent fails many times, but usually recovers.

Figure 9.1 shows that the agent learns to complete the task more quickly over several episodes. The dotted line shows the approximately optimal number of decision cycles. The line is approximate because determining the actual optimal is very difficult since it is not the lowest possible number of decision cycles, but rather corresponds to the highest possible reward, which is difficult to determine analytically. Figure 9.2 shows that the total reward also increases across those episodes. Again, the approximate optimal is shown. The implication is that the agent is in fact learning to do the task better. Figure 9.3 shows that the total number of failures across trials is low early on (before much learning takes place), but gets worse in the middle and then better at the end This implies that the agent is learning cyclical behavior, but generally recovering. Table 9.2 shows that the agent did experience several Total Failures across 12 of the 50 trials (24%), but only 3 trials actually ended in failure, which corroborates the conclusion above that the agent recovered most of the time. This implies that the agent would

probably do better (in terms of failures) if the cutoff were raised; however, this would not affect the results for the number of decision cycles and total reward, so we did not run additional experiments.

We conclude from these results that the Conduciveness appraisal is sufficient for learning, although there is certainly room for improvement.

9.6 Exploring Outcome Probability and Discrepancy from Expectation

As described before, making predictions is a central aspect of the model. In Scherer's theory, predictions are associated with an Outcome Probability appraisal, and their success or failure is captured via the Discrepancy from Expectation appraisal. Thus, the inclusion of these makes sense for our model. The inclusion should give us insights into how these appraisals influence learning.

We expect these appraisals to influence feeling intensity (and thus reward, since it is calculated from feeling intensity). As discussed earlier, an outcome that is expected should result in less intense emotions than one that is unexpected. This principle is integrated into our feeling intensity equation via the Outcome Probability and Discrepancy from Expectation appraisals, making them crucial to our theory. Furthermore, since feeling intensity is an input to the calculation of reward, these dimensions will influence the reward the agent gets, and hence its learning.

Generating values for these requires generating predictions. In the previous model, prediction generation was very simple: the agent always predicted that the next stimulus would be passable and on the path, except when it was about to accomplish the task, in which case it predicted that the next stimulus would be the task completion stimulus.

In an effort to make more realistic predictions, we introduced a new memory. This memory records every stimulus that the agent experiences and links representing which stimuli have ever occurred in sequence, making it conceptually similar to Nuxoll's (2007) episodic memory, but our memory is more tailored to our needs. Specifically, it

also includes "strength" values for each link, which is intended to represent how frequently and recently a particular sequence occurred. The frequency and recency information tells the agent how commonly this link is used. The idea is that, as it gains experience, the agent will settle into following the same sequence of actions (namely, those that maximize reward), thus experience the same sequence of stimuli. This will cause the strengths of the links between those stimuli to increase (and the strengths of other links to decrease). The agent can then use this information to make predictions about what it thinks will happen next (e.g., the strongest link connects to the stimulus that has occurred next most commonly). The use of strength perhaps makes it more similar to Kaplan & Kaplan's (1982) notion of contiguity in cognitive maps (i.e., stimuli that occur close together in time tend to become associated). We will simply refer to this memory as the episodic memory (not to be confused with Nuxoll's episodic memory).

The strengths of the links in the episodic memory have values in the [0,1] range and are updated as follows. When a stimulus is attended to, the strength of the link between the previously attended stimulus and the next stimulus is increased by a constant (we used 0.2 because it allowed the agent to potentially maximize the strength in a fairly small number of experiences, but not so small that it would thrash). If this is a new link, then it is given an initial value of that constant. To preserve contiguity, other links that are not used must be decreased in strength (otherwise, everything could eventually be associated with everything). Thus, all other links emanating from the previous stimulus are decreased in strength by another constant (we used 0.05, which is smaller than our increase constant, to reflect that the decrease is probably being "split" among many links, as per Kaplan et al. (1991). A better model might actually calculate the decrease based on the number of links, but we started with a simple model that proved sufficient for our purposes). Strength values are capped at 0 and 1. Thus, if the agent learns to do some particular sequence, then that link will eventually have strength 1 while all other links from the previous stimulus will have strength 0. Finally, each recorded stimulus has a total strength value, which is the sum of the strengths of the links emanating from it. This value is used to generate a pseudo probability as described below.

The stimuli that are recorded in the episodic memory are exactly those stimuli that the agent encodes, but augmented with a context value. The purpose of the context is to disambiguate some stimuli so the agent can learn the sequences properly. For example, the current stimulus might be gateway-17, but unless the agent records which side of gateway-17 it is on, the next stimulus in the sequence will be pulled between the stimuli for each of the rooms. The context value is actually a combination of the current room, the current task type, and the current task object id. By including this information, most sequences can be disambiguated, which essentially reduces the noise in the system, giving us to get a clearer picture of what the agent is able to learn.

This episodic memory is used to generate values for the Outcome Probability and Discrepancy from Expectation appraisals in the following way. When the agent Intends something, it creates a prediction with an associated Outcome Probability. The prediction is the stimulus at the end of the strongest link, and the Outcome Probability is the strength of that link divided by the previous stimulus's total strength. If there are no links (because the previous stimulus is new), then the agent makes a default "passable, on path" prediction like the original model, with an Outcome Probability of 0.5. Discrepancy from Expectation is calculated as 0 if the currently Attended stimulus matches the prediction, and 0.5 if not (as determined by Comprehend).

While this model of the Outcome Probability and Discrepancy from Expectation appraisals is an improvement over the original model's methods, there are issues. First, the value calculated for Outcome Probability is not actually a probability. Rather, it is pseudo probability in the sense that, the larger it is, the more likely the event is to occur. This follows directly from the way in which the strengths are computed. Certainly, at extreme values, the agent must have a history of almost always or almost never following that link, and thus the value most likely does represent a probability in those cases. Again, the value reflects frequency and recency, so even if, historically, the agent has followed link A more, if recently it has repeatedly followed link B, then the stimulus corresponding to link B will be given a high Outcome Probability.

Second, one could argue that predictions should be based on actions instead of stimuli (i.e., predicting the outcome of actions taken in some state). However, since

actions are tied to stimuli in this model, this is really the same thing. Furthermore, the agent is not predicting the outcome of an action/stimulus, but rather what it will be Attending to next. These are related in that the situation the agent ends up in as a result of the action it takes contains many possible stimuli the agent can Attend to, one of which should be the predicted stimulus. But the agent may still choose to Attend to some other stimulus instead. In other words, the agent is not predicting the situation it will end up in, but which stimulus it will Attend to in that situation. Finally, one could argue that when there is a mismatch between the prediction and reality, the value of Discrepancy from Expectation should reflect the extent of the mismatch. For example, the prediction is partially correct (e.g., it correctly predicted a gateway, but the wrong one), the Discrepancy value should be lower than if the prediction is completely off. We agree that this is worth exploring in the future, but felt our simpler model was a sufficient starting point for our exploration. Thus, there is plenty of room for improvement in this model, but our results should be indicative of the usefulness of these appraisals.

Finally, we expect the way in which these appraisals are generated to have some side effects: reward will tend towards zero as the agent accumulates experience, which should help eliminate the infinite reward cycles described earlier. As the agent learns, it will settle into a fixed sequence. Repeating this sequence enough will result in the strengths of the associated links going to 1, while all other strengths go to 0. Thus, these stimuli will be predicted with an Outcome Probability of 1. Since the agent is following this sequence, it will predict the associated stimuli and thus Discrepancy from Expectation will be 0. With these values, the calculated intensity will be 0, and thus the reward will be 0. Thus, we expect total accumulated reward to go up in earlier episodes, but down in later episodes. In other words, the reward becomes non-stationary. In the case of an infinite reward cycle, this may eventually break the cycle as the values of the Attend operators are reduced towards 0 (but if the alternative values are negative, a cycle may still exist). In practice, such cycles are very rare, so the agent doesn't usually find them.

9.6.1 Outcome Probability and Discrepancy from Expectation Results

The experimental conditions are exactly as reported earlier. The Conduciveness appraisal is still enabled, as are the Outcome Probability and Discrepancy from Expectation appraisals.

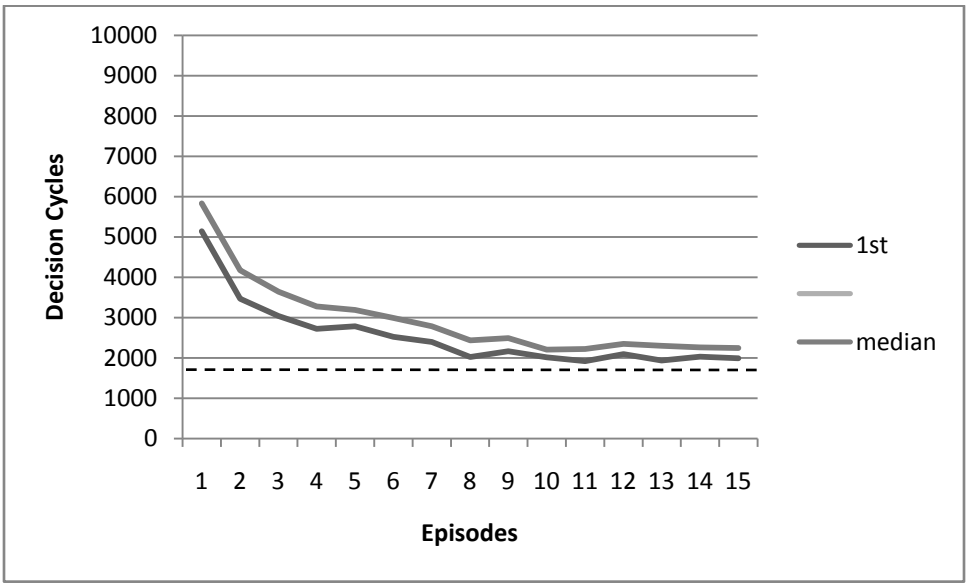


Figure 9.4: Decision cycles to task completion for Outcome Probability and Discrepancy from Expectation experiment.
The agent learns to complete the task in fewer decision cycles across several episodes. Median, first, and third quartiles shown.

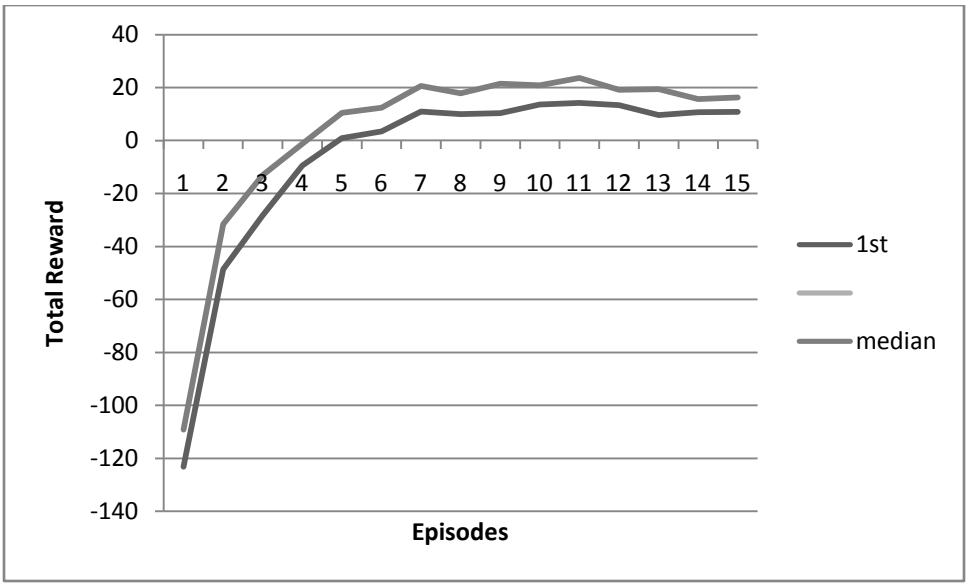


Figure 9.5: Total reward accumulated during task for Outcome Probability and Discrepancy from Expectation experiment.
The agent learns to accumulate more total reward across several episodes, but reward decreases slightly towards the end.

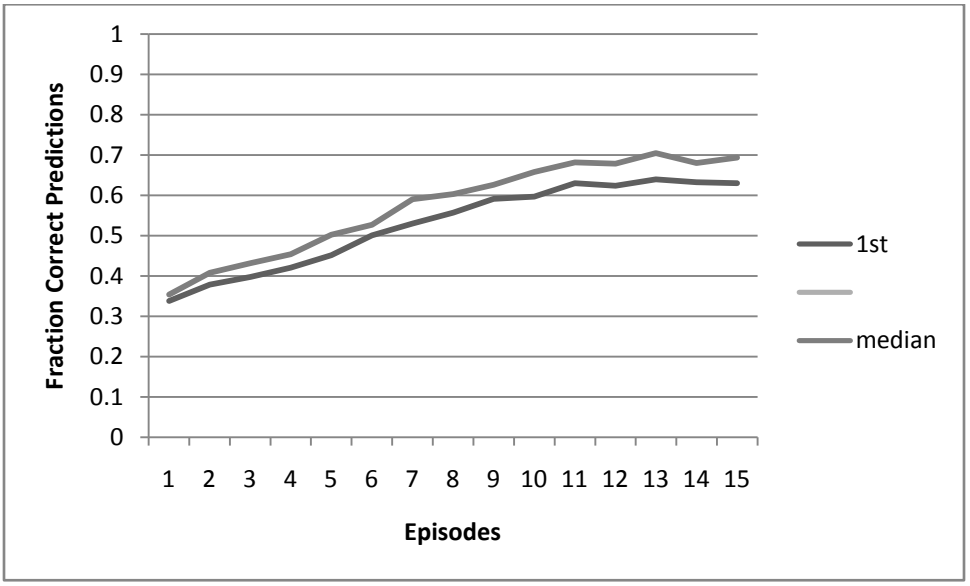


Figure 9.6: Fraction of predictions made that are correct for Outcome Probability and Discrepancy from Expectation experiment.
The agent learns to predict more accurately across several episodes.

Final Episode Failures	Trial Failures	Total Failures
0	0	0

Table 9.3: Failures for Outcome Probability and Discrepancy from Expectation experiment. The agent never fails.

Figure 9.4 shows that the agent learns to complete the task in fewer decision cycles over several episodes. Figure 9.5 shows that the agent also learns to increase its total reward, but with a slight decrease towards the end. This matches our qualitative prediction that reward would go down. The approximate optimal reward is not shown here since reward is non-stationary and thus does not tend towards some maximum. This will be true for the remainder of the reward results in this chapter. The agent's ability to correctly predict outcomes also improved across episodes, as shown in Figure 9.6. Thus, the episodic memory and the appraisals seem to be functioning as intended.

Comparing to the agent with Conduciveness only, the number of decision cycles did not change (compare Figure 9.1 and Figure 9.4). However, the number of failures went to 0 across all categories (Table 9.3), implying that cycles were broken, or that the agent learned good behavior more quickly. (Note that good behavior does not necessarily correspond to fewer decision cycles). Finally, we note that the scale of the rewards is reduced compared to the Conduciveness only condition (Figure 9.2). Again, this is expected because the intensity's "surprise factor" will often be less than 1 (and never more than 1), forcing the intensity (and thus reward) to have a smaller magnitude in general.

We conclude from these results that adding the Outcome Probability and Discrepancy from Expectation appraisals to the system results in learning improvements via the ability to make better predictions and to generate more accurate values for these appraisals with respect to those predictions.

9.7 Exploring Goal Relevance

The purpose of Goal Relevance is to help the agent choose what to Attend to. Specifically, if something is not Goal Relevant, the agent shouldn't waste any resources on it. By including this, we hope to learn how a proactive influence on what stimulus to

Attend to interacts with reinforcement learning (which learns after an action has been selected).

In the revised model, Goal Relevance is entirely based on the stimulus's path on/off augmentation, as shown in Table 9.4. In other words, Goal Relevance is the continuous numeric translation of the binary values for path. (Clearly, in a more complex model, it may be more that that).

Path On	Path Off
1.0	0.0

Table 9.4: Goal Relevance values.

There are multiple ways in which Goal Relevance could influence what the agent attends to. The most obvious is simply to include allow it to influence feeling intensity, as before. Thus, more relevant stimuli will result in more intense feelings (and thus more extreme rewards). One problem with this approach is that, if there are many other appraisals, the effect of Goal Relevance will be washed out (since, unlike Outcome Probability and Discrepancy from Expectation, it is just averaged in with the others). On the other hand, if there are very few other appraisals, Goal Relevance can skew the values too strongly. Indeed, in experiments not shown, allowing Goal Relevance to influence feeling intensity resulted in the creation of infinite reward cycles. The problem was that, for positive rewards, Conduciveness and Goal Relevance were both 1 (because the stimulus was on path), so the magnitude of positive rewards was unchanged. But for negative rewards in the case where the stimulus was on path but progress was false, Conduciveness would be -0.5 while Goal Relevance was still 1. The effect was to essentially decrease the magnitude of negative rewards in enough cases that the agent was able to nearly always find an infinite reward cycle. We could possibly have avoided this by manipulating the exact values of the appraisals, or the underlying values from which they are generated, but this was undesirable since it meant exploring a large parameter space, and probably finding a domain-specific set of values.

An alternative solution is to recognize that, by giving a stimulus a particular path value, the agent is indicating that it already knows if a stimulus is relevant or not. Thus,

rather than just learning after the fact that a stimulus should or should not have been Attended to, the agent can influence the selection directly.

We call this direct influence value boosting, which works like this. Each Attend operator has some value as learned by the agent via reinforcement learning. The agent "boosts" this value by adding the value of the Goal Relevance appraisal to it. Thus, for stimuli on the path, the agent adds 1 to the value, and for stimuli off the path, the value is unchanged (0 is added). Since the rewards the agent gets are no greater than 1, most of the values are less than 1, and thus adding 1 is a major boost. Essentially, the agent ignores stimuli that are off the path.

This does not mean that the agent is automatically perfect. In any given situation, there may be many stimuli that are on the path. For example, if the agent is in the supertask and not carrying a block, all subtasks to go to rooms that have not yet been visited, or that are known to contain blocks, are on the path. The agent must still learn which order to do these in. Furthermore, because the value of these operators is artificially inflated, the learned values will often be negative (to reduce the summed value closer to the actual value). Given enough time, the value of an operator can actually be reduced to the point that other operators can have higher values.

9.7.1 Goal Relevance Results

The experimental conditions are exactly as reported earlier. The Conduciveness Outcome Probability and Discrepancy from Expectation appraisals are still enabled. The Goal Relevance appraisal does not influence the feeling intensity or reward directly, but via value boosting.

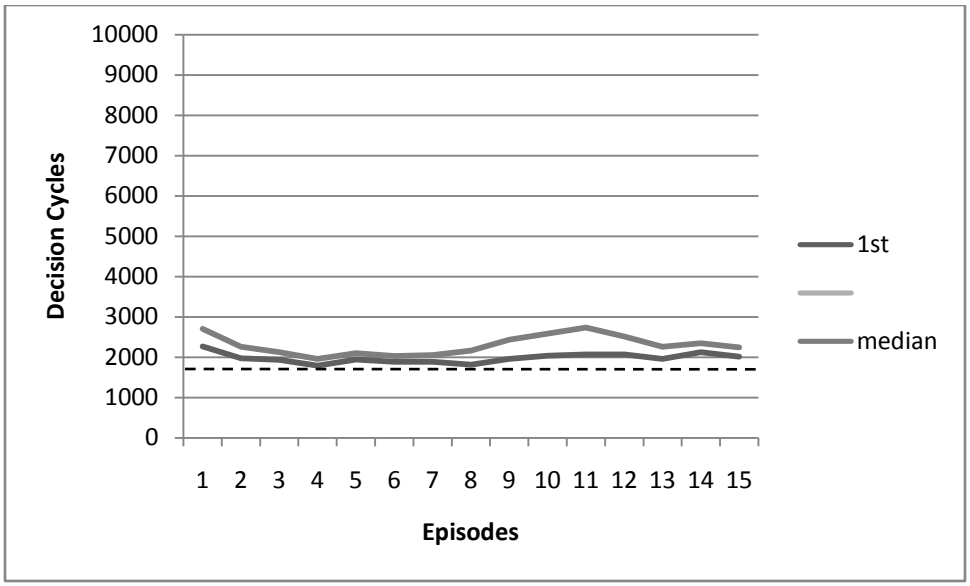


Figure 9.7: Decision cycles to task completion for Goal Relevance experiment. Except for a blip in the third quartile and some minor learning early on, the agent's performance is flat.

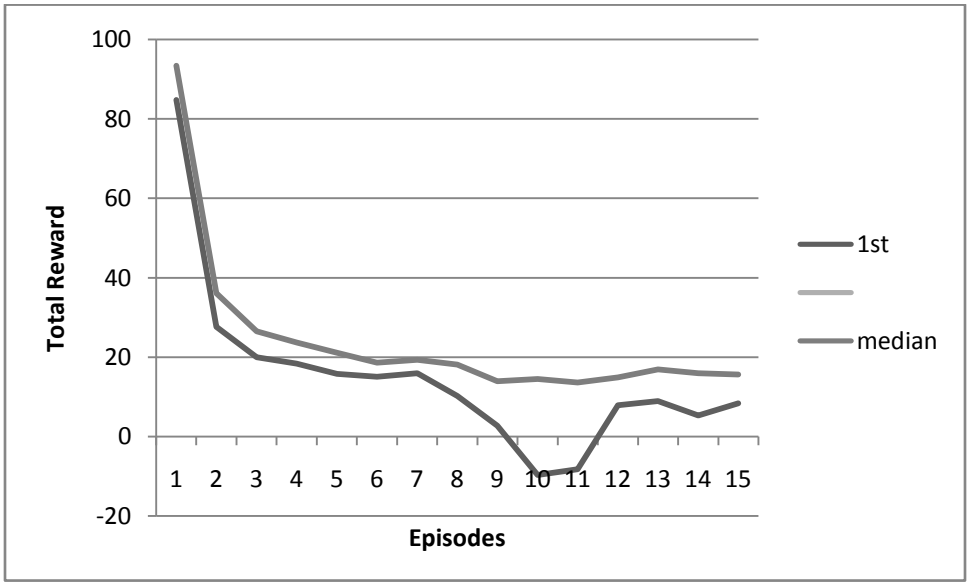


Figure 9.8: Total reward accumulated during task for Goal Relevance experiment. Reward decreases across episodes, with an intermediate pronounced dip corresponding to the pronounced rise in decision cycles.

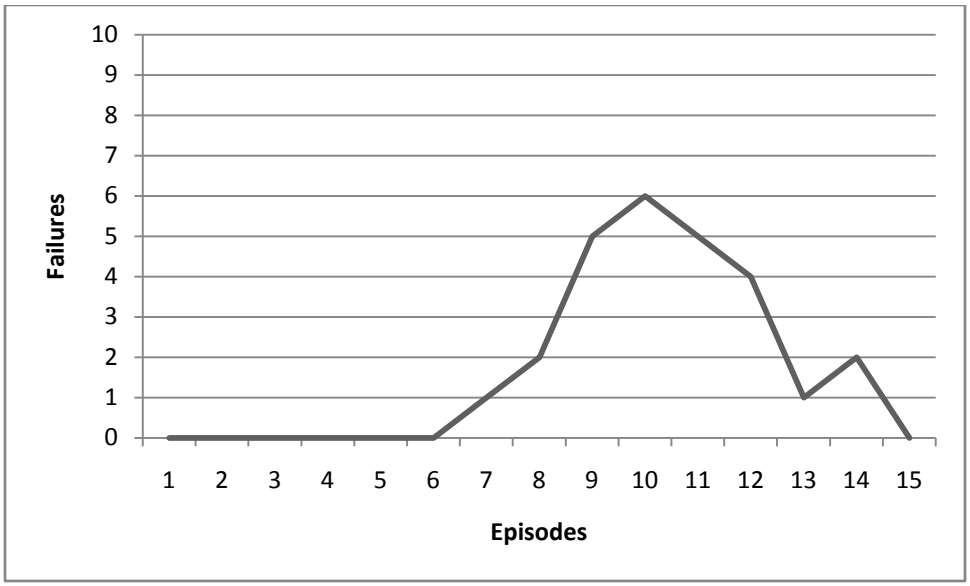


Figure 9.9: Failures across trials for each episode for Goal Relevance experiment. There is a pronounced peak corresponding to the pronounced variations in the other data.

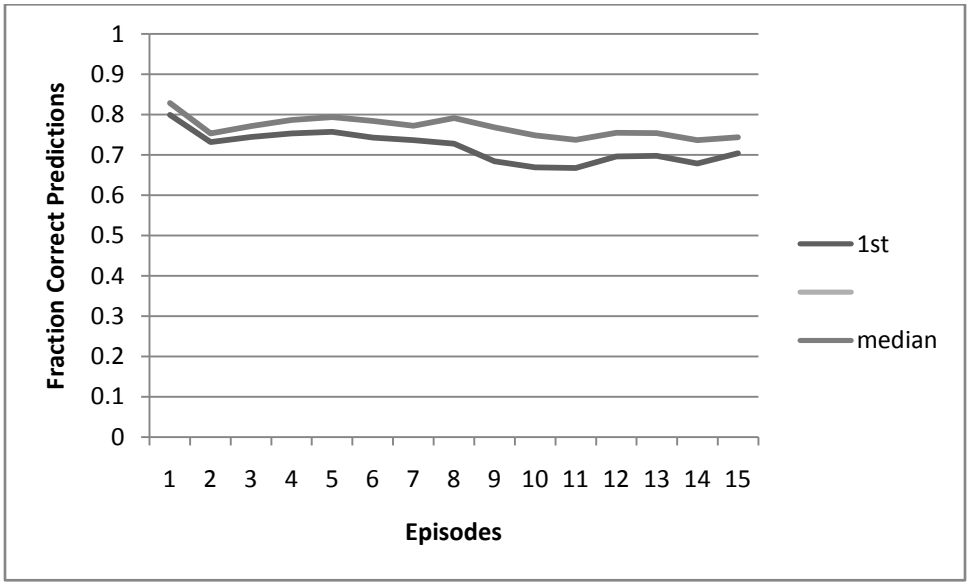


Figure 9.10: Fraction of predictions made that are correct for Goal Relevance experiment. The agent's predictions are fairly flat, ending slightly worse than how it began.

Final Episode Failures	Trial Failures	Total Failures
0	12	26

Table 9.5: Failures for Goal Relevance experiment. The agent fails many times, but always recovers.

Except for a small amount of early learning and a blip in the middle, the number of decision cycles to task completion is flat (Figure 9.7). This is to be expected, given that the agent knows the path value for every stimulus; all it must learn is a good order in which to Attend to the on path stimuli (which accounts for the early learning). The blip occurs because, as described earlier, the boosting has artificially inflated the values of some operators, causing the learned values to become very negative. In some cases, these values will become negative enough that they outweigh the boost, which may suddenly cause some other stimuli (which are probably off path and untried) to become higher-valued. The agent will then Attend to these stimuli until it discovers that they are even worse, at which point it returns to the on path stimuli.

The reward starts high and goes low (Figure 9.8). Again, this is expected, as the agent is choosing stimuli that are on path, and thus highly rewarding, since the agent has not experienced them much yet. As it repeats these choices, however, the surprise factor of the intensity calculation decreases, leading to decreased reward. There is also blip corresponding to the one in the decision cycles.

Similar to the decision cycles, the fraction of correct predictions is fairly flat (Figure 9.10). There is a slight initial decrease resulting from the fact that, early on, the episodic memory contains no information, so the agent is always making the default prediction (passable and on path). Since this prediction is very vague, and the agent is choosing nearly all on path stimuli (except for exploratory moves), the predictions are correct the vast majority of the time. As the episodic memory grows, the agent makes more specific predictions, and thus is wrong more often.

The failure data (Figure 9.9 and Table 9.5) show a peak in failures around episode 10, corresponding to blips in the decision cycle and reward data. This is consistent with the explanations for those above.

9.7.2 Knowledge Revision

We found these results somewhat uncompelling. On the one hand, value boosting works. However, due to the exhaustive knowledge the agent has regarding path information, it looks like it accounts for nearly all of the agent's performance, which

raises questions of the usefulness of the other appraisals. Does this mean that the other appraisals are unnecessary? The answer is no. First, there is some learning going on, even if only a little. Furthermore, Goal Relevance alone cannot be used to generate rewards in this model since it does not have a valence. Thus, Conduciveness, at least, is necessary. Additionally, the number of failures is higher than it is when Goal Relevance is not included, yet the agent still always finishes the task. This implies that there is indeed learning going on that allows the agent to recover, and as we saw, Outcome Probability and Discrepancy from Expectation have positive effects on failure reduction.

Still, the results would be more compelling if the learning was more obvious. Given the complexity of the task, the agent needs a lot of this kind of domain-specific knowledge or else it fails to learn (experiments not shown). However, we may be able to reduce the agent's knowledge some, and thus provide greater opportunity for learning, thus better demonstrating how the appraisals can all work together. That is, we want to show that, even in the absence of exhaustive knowledge, this appraisal still has a positive influence on the agent's learning.

Performing an exhaustive set of experiments to determine the minimum amount of acceptable knowledge would have been prohibitively expensive. Instead, we pulled out a few key pieces of knowledge and made the path value for those situations "unknown". This had the side effect of requiring us to introduce an "unknown" value for progress as well. This is because, when the agent first creates a subtask (and thus has no reference point to compare progress in that subtask), it inherits the path value of the subtask stimulus itself (so if the task was on path, then the first step is considered to be making progress, etc.). Thus, if the path value for the task is unknown, then the progress for the first step in a subtask will also be unknown. Thus, we expanded the definitions of Goal Conduciveness and Goal Relevance to account for these new values. For Conduciveness, only the clearly good situation is given a positive value; the others are negative to discourage cycles (Table 9.6). For Goal Relevance, unknown is given an intermediate value (Table 9.7).

	Path: On	Path: Unknown	Path Off
Progress: True	1.0	-0.25	-0.5
Progress: Unknown	-0.25	-0.25	-0.75
Progress: False	-0.5	-0.75	-1.0

Table 9.6: Conduciveness values (accounting for unknown path and progress). The values for the old cases are unchanged.

Path On	Path Unknown	Path Off
1.0	0.5	0.0

Table 9.7: Goal Relevance values (accounting for unknown path). The old values for the old cases are unchanged.

One piece of knowledge removed concerned the go to room subtask. When the agent is in room A and wants to get to room B, and these rooms are adjacent, then the stimulus corresponding to the gateway connecting the rooms is considered on path, and all other gateways are off path. This knowledge was left in. The knowledge that was removed covered the situation when the rooms were not adjacent. For that case, previously the agent would look in the task-specific map that it had created and find which adjacent rooms were closer to the destination than the current room. Gateways leading to those rooms were marked as on path. Now, if the agent is more than one room away from the destination, the path values for the gateways are all unknown, and it must learn the best route to the destination.

The other piece of knowledge concerned when to attend to rooms when in the clean house supertask. The agent usually wants to create subtasks when in the supertask, and always wants to ignore gateways and blocks (if the agent wants to go somewhere or clean something, it should create the appropriate subtask). This knowledge was left in. With regards to attending to rooms while in the supertask, however, the situation is more complex. Usually the agent wants to ignore rooms just as it wants to ignore gateways and blocks. The exception is when the agent is carrying a block and is in the storage room. In that case, Attending to the room results in the agent putting down the block. The knowledge about when room stimuli in the supertask are on or off the path was removed and made unknown.

9.7.3 Reduced Knowledge Results

For the reduced knowledge version, we ran for 30 episodes instead of 15, and set the cutoff at 20000 decision cycles instead of 10000. This is to help the agent cope with the increased difficulty of the task.

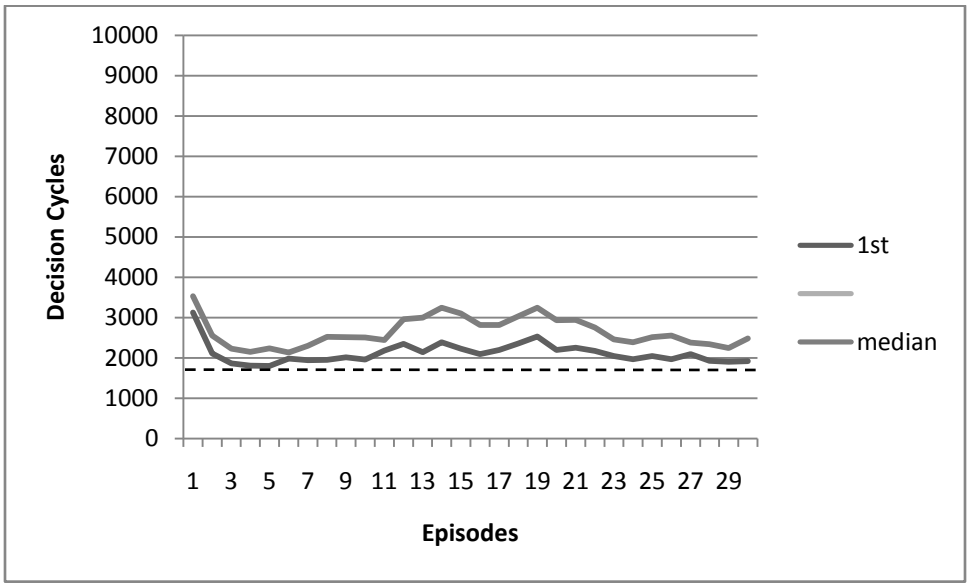


Figure 9.11: Decision cycles to task completion for reduced knowledge Goal Relevance experiment. The agent learns at first, followed by some unlearning and recovery.

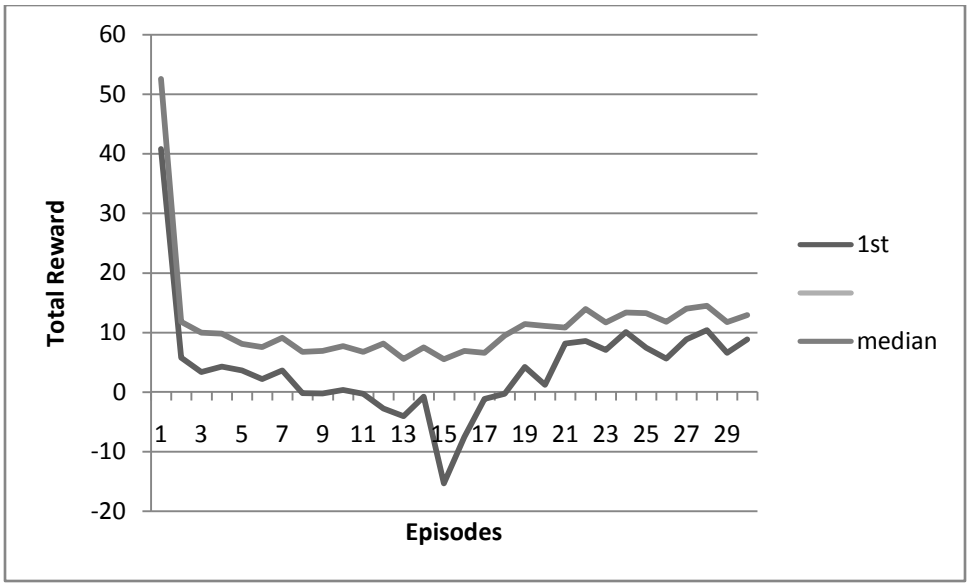


Figure 9.12: Total reward accumulated during task for reduced knowledge Goal Relevance experiment.
Reward roughly tracks the shape of the decision cycles.

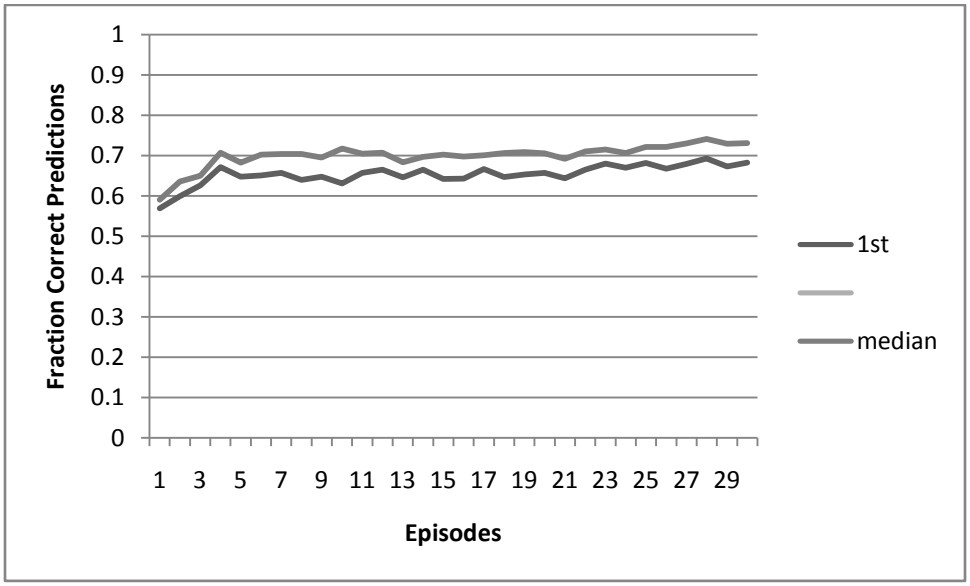


Figure 9.13: Fraction of predictions made that are correct for reduced knowledge Goal Relevance experiment.
Predictions get better at first and then level off.

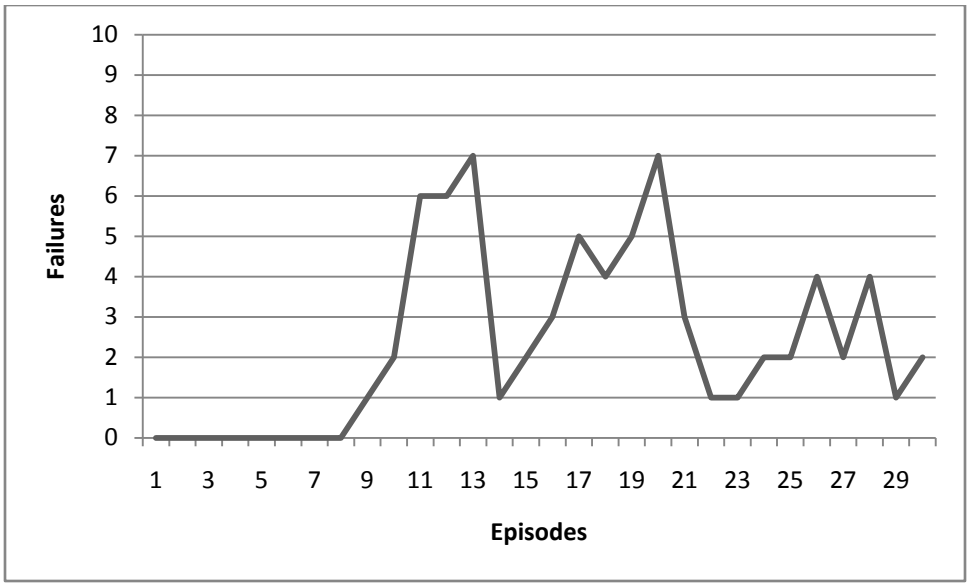


Figure 9.14: Failures across trials for each episode for reduced knowledge Goal Relevance experiment.
Failures are erratic after initial learning, but reduced towards end.

Final Episode Failures	Trial Failures	Total Failures
2	29	58

Table 9.8: Failures for reduced knowledge Goal Relevance experiment.
The agent fails many times, but usually recovers.

The reduced knowledge results show that the agent must still learn; Goal Relevance does not eliminate the usefulness of the other appraisals. There are at least two kinds of learning that are happening: early and late. The early learning takes place in the first 3-5 episodes (see all figures above), and is much more pronounced than in the agent with full knowledge. It is here that the agent is learning what to do with the "unknown" stimuli. After that, the agent experiences regression in learning (most pronounced in Figure 9.11 and Figure 9.14; note the correspondence between the third quartile in Figure 9.11, the first quartile in Figure 9.12, and the peaks in Figure 9.14), just as it did in the full knowledge case (Figure 9.7). In this case, the regression is worse, probably because some of the stimuli the agent gets exposed to are "unknown". Thus, whereas before the agent only had to deal with "off path" stimuli, it now must also deal with "unknown" stimuli. Still, the agent eventually recovers.

We conclude that Goal Relevance has a generally positive impact on performance by reducing the number of contenders for Attention that the agent must learn to distinguish among. With exhaustive knowledge, the agent is able to do this very well, but even with reduced knowledge the agent generally does better than without this appraisal. This implies that, in addition to Goal Relevance, the other Relevance appraisals (Suddenness, etc.) are also worth future exploration.

9.8 Exploring Intrinsic Pleasantness

Intrinsic Pleasantness is an unusual appraisal because its value is independent of the current task. Another way to look at this is that it reflects long-term learning or an evolutionary adaptation to treat certain stimuli as always good or bad. For example, someone on a diet may still find cake to be Intrinsically Pleasant because of the rewards it has consistently provided in the past (or because evolution has made us inherently desire high-energy foods). Our model does not attempt to learn values for Intrinsic Pleasantness, but it does allow us to explore the effects of Intrinsic Pleasantness values for various stimuli on behavior and learning.

As with Goal Relevance, we can either just allow Intrinsic Pleasantness to influence intensity, or we can use value boosting. Unlike Goal Relevance, as a valenced appraisal, Intrinsic Pleasantness also influences valence (until now, valence was entirely determined by Conduciveness). This makes intuitive sense (consider the cake example), and is what motivated us to make the circumplex-inspired changes described earlier. Additionally, Intrinsic Pleasantness does not share underlying values with other appraisals like Goal Relevance did, so we don't expect there to be inherent conflict like there was between Goal Relevance and Conduciveness. Finally, value boosting for Intrinsic Pleasantness requires the development of additional theory; for example, what do we do with negative values? Do we still allow Intrinsic Pleasantness to influence valence but not intensity? Or do we allow both? Thus, to keep things simple, we decided to explore Intrinsic Pleasantness without value boosting.

9.8.1 Intrinsic Pleasantness Results

The experimental conditions are exactly as reported earlier, with a cutoff of 20000 and 15 episodes. The Conduciveness Outcome Probability and Discrepancy from Expectation appraisals are still enabled, but Goal Relevance is not. Intrinsic Pleasantness influences feeling intensity and valence as described earlier.

For this experiment, we gave blocks an Intrinsic Pleasantness value of 1.0 (everything else was neutral). This means that the valence generated for a block stimulus will have a strong positive bias. This will interact with the Conduciveness value, which will either reinforce that bias, or counteract it. Depending on the situation, this may have positive or negative effects on the agent's learning and behavior. For example, in many cases, Attending to a block will ultimately be a good thing, as this is necessary in order to get blocks into the storage room. In these cases, boosting the valence of the block will help the agent learn more quickly that this is a good thing to do. However, there are other cases where Attending to a block is the wrong thing to do. For example, if the agent is already carrying a block, or if the block is in the storage room, then Attending to the block is a distraction. When Conduciveness was the only input to valence, the block would have had a negative valence in these situations. However, with Intrinsic Pleasantness, these may end up having a positive or neutral valence. This may make it more difficult for the agent to learn not to do those things. In fact, there may be new reward cycles because of the increase in positive valence. If so, the agent may still be able to learn to overcome these at the cost of slower learning.

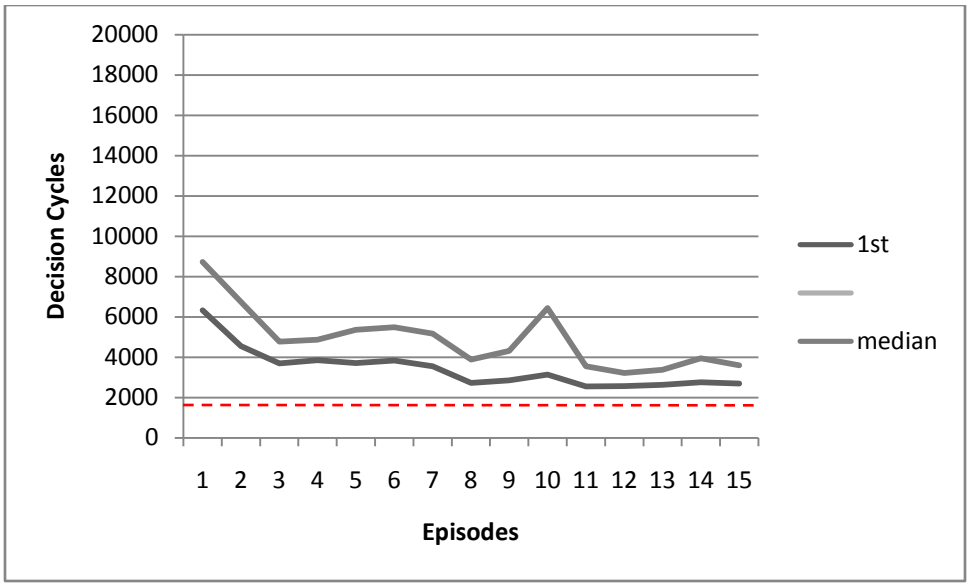


Figure 9.15: Decision cycles to task completion for Intrinsic Pleasantness experiment. The agent learns across episodes, but doesn't do as well in the third quartile.

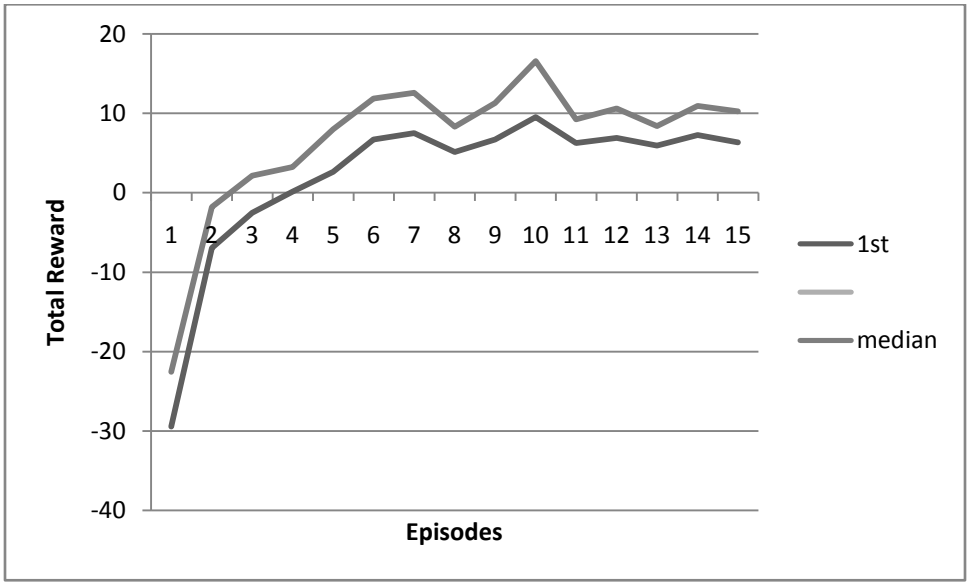


Figure 9.16: Total reward accumulated during task for Intrinsic Pleasantness experiment. The agent learns to get more reward across episodes.
The third quartile here appears to correspond to the third quartile in the decision cycles, implying that there were some reward cycles.

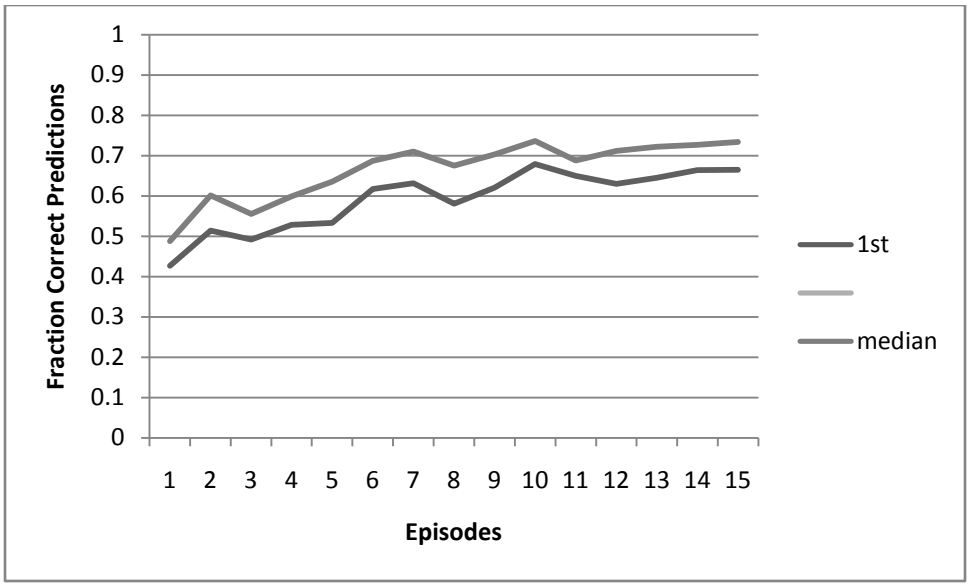


Figure 9.17: Fraction of predictions made that are correct for Intrinsic Pleasantness experiment. The agent learns to improve its predictions across episodes.

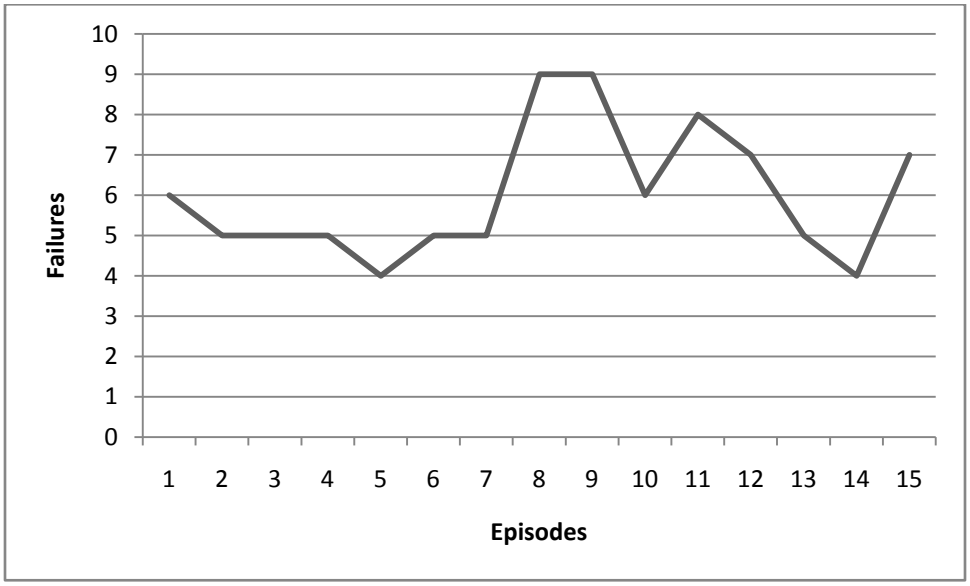


Figure 9.18: Failures across trials for each episode for Intrinsic Pleasantness experiment. Failures remain fairly flat.

Final Episode Failures	Trial Failures	Total Failures
7	23	90

Table 9.9: Failures for Intrinsic Pleasantness experiment.
The agent had at least one failure in almost half of the trials, but was able to recover most of the time.

In terms of decision cycles (Figure 9.15), the agent does fairly well in the first two quartiles, but not the third. The third quartile probably represents cases in which the agent had more trouble with conflicted events (where Conduciveness and Intrinsic Pleasantness have conflicting values). Even in the first two quartiles, the agent does less well in comparison to an agent without Intrinsic Pleasantness (Figure 9.4); it starts and finishes higher (note the scales on the two graphs are different). This is to be expected, though, given that Intrinsic Pleasantness can sometimes conflict with Conduciveness. The total reward (Figure 9.16) tells a similar story. The third quartile seems to indicate that the agent found some reward cycles (hence the spike in the middle), which would explain why the third quartile for the decision cycles is so much higher. The agent also learns to make better predictions (Figure 9.17). Compared to an agent without Intrinsic Pleasantness (Figure 9.6), the agent actually starts better but ends worse. The early success could be a reflection of early cyclical behavior (i.e., the agent learning to predict bad cycles) or that the agent learns to pick up blocks faster. The slightly worse prediction performance at the end is just a reflection of the conflict between Intrinsic Pleasantness and Conduciveness. Finally, the agent has many failures (Figure 9.18 and Table 9.9), but learns to recover from most of them at the very end. Thus, while there were cycles the agent was able to learn to overcome them. That is, they were not infinite in that negative reward from later stimuli eventually overcame the positive bias. Furthermore, the repetition inherent in cyclical behavior would have reduced the feeling intensity due to the interaction between Outcome Probability, Discrepancy from Expectation, and the episodic memory.

Thus, we see an interesting interaction between Intrinsic Pleasantness and the task—they can sometimes conflict, but learning is able to overcome these issues. In other tasks, it may be that Intrinsic Pleasantness is always helpful or always conflicting, in which case we would expect to see more extreme results (either good or bad). However, in order for this appraisal to make functional sense, the net effect should be positive. Thus, further exploration into this appraisal should be conducted in the future.

9.9 Additional Experiments

There were at least two experiments we wanted to conduct with the revised system that are not directly related to specific appraisals; the purpose of these experiments was to explore extensions to the model. First, we wanted to explore whether mood has the same positive effect here as it did in the Eaters domain. Second, in all of these experiments so far, the learning and exploration rates were constant. This may have prevented the system from achieving its full potential because it may not have been able to converge properly, and exploration may have forced it to make some bad moves. We explored whether emotion can be used to modulate these aspects of reinforcement learning.

9.9.1 Mood

We tested mood with Conduciveness, Outcome Probability, and Discrepancy from Expectation enabled, with many different values for the decay and movement rates (results not shown). We hypothesized that mood would essentially propagate values forward to new states, giving them better initial values. Unfortunately, mood seemed to have virtually no impact on the agent (results not shown); the results are nearly identical to the agent under the same conditions without mood. The only difference is that the scale of the total rewards is much larger (the max is higher and the min is lower). This is not surprising because, with mood, the agent is getting non-zero rewards all the time, not just in the couple decision cycles between Attend and Intend. Still, the shape of the rewards is the same.

We speculate that the reason mood has no impact in this domain may be because of the domain itself, or the reduced noise (e.g., fewer and higher quality appraisals), or possibly an unexpected impact of one of the many changes we made to the agent. While this is essentially a negative result, it is also worth noting that mood did not have a negative impact on the agent. Clearly, further exploration is required here in the future.

9.9.2 Dynamic Exploration Rate

Traditional reinforcement learning systems usually decrease exploration rate in a fixed way (e.g., linear across actions or episodes). While this approach may work for

agents in fixed domains, we find this approach unsatisfactory for a system that aspires to be general, since the agent can never know how many actions or episodes it has left for any given task.

Wilson (1996) describes several exploration strategies, grouping them into global strategies (e.g., constant rate, descending function of time, or statistical measurements of properties like reward or error over time) and local strategies (e.g., statistical measurements of properties relating to the current input). Work by Hogewoning (2007), mentioned earlier, is a global strategy that automatically adjusts exploration rate based on statistical analysis of long-term and short-term rewards over time. This strategy requires keeping a history of rewards. Furthermore, they assume the agent is using Boltzmann action selection, whereas we are using epsilon greedy.

We tried a couple of simpler approaches based on the agent's feeling intensity and valence. The idea is that if the feeling valence is positive, then things are probably going well, so exploration rate is set to 0. If the feeling valence is negative, then things could probably be going better, so the exploration rate is set to the absolute value of the reward (feeling intensity multiplied by feeling valence). That is, the worse things are (in terms of both intensity and valence), the higher the exploration rate. The exploration rate naturally falls in the [0,1] range with this setup. If mood is disabled, this is a local exploration strategy (since it is determined by the current stimulus only). If mood is enabled, then this is essentially a hybrid global/local strategy since the agent's current feeling is a combination of its emotion (which is about the current stimulus) and its mood (which is about recent stimuli).

Below we show the results with emotion only. Conduciveness, Outcome Probability and Discrepancy from Expectation are enabled. The cutoff is set at 10000 decision cycles (so these results are directly comparable to those in Figure 9.4, Figure 9.5, Figure 9.6 and Table 9.3).

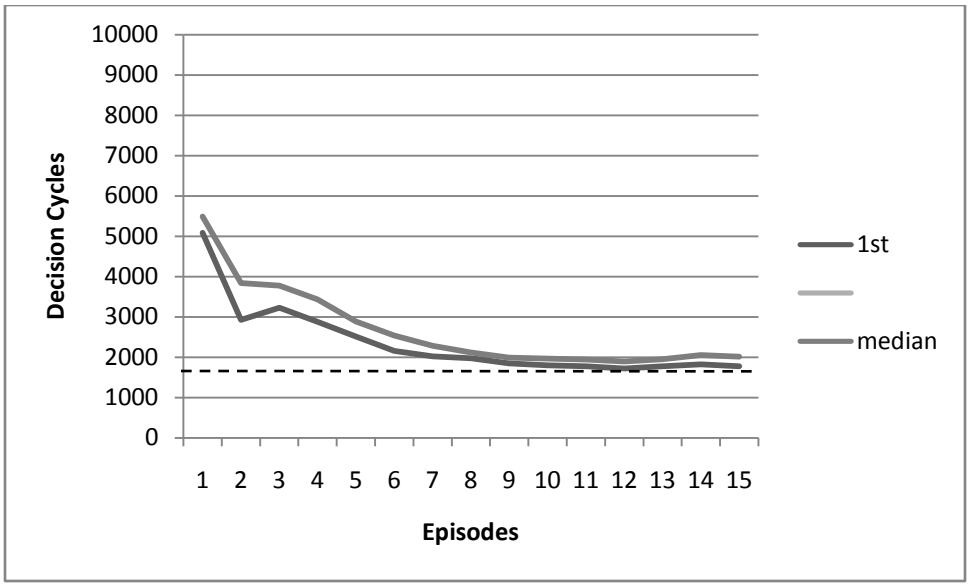


Figure 9.19: Decision cycles to task completion for Dynamic Exploration experiment. Compared to the agent without Dynamic Exploration, this agent takes slightly fewer decision cycles.

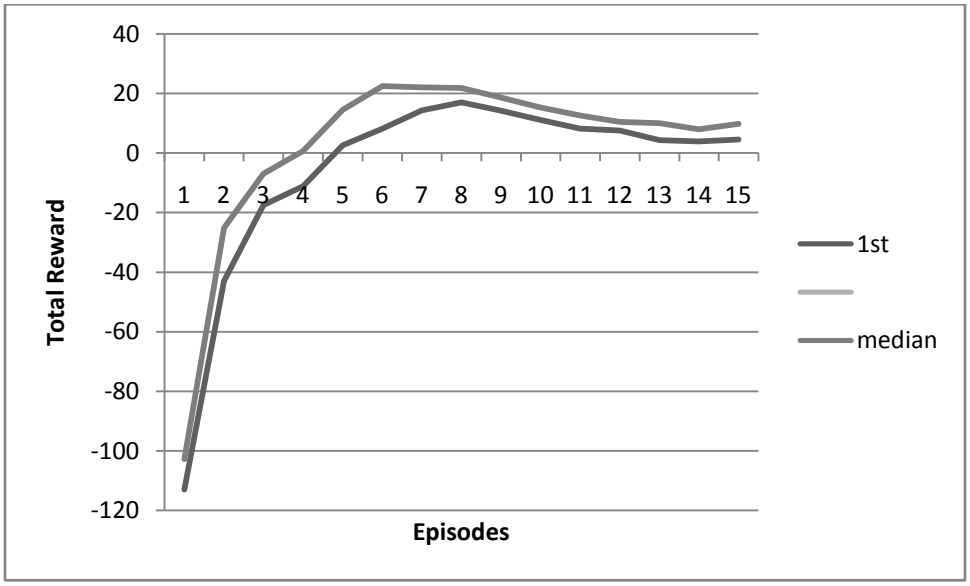


Figure 9.20: Total reward accumulated during task for Dynamic Exploration experiment. Compared to the agent without Dynamic Exploration, this agent has a much more pronounced dip in reward.

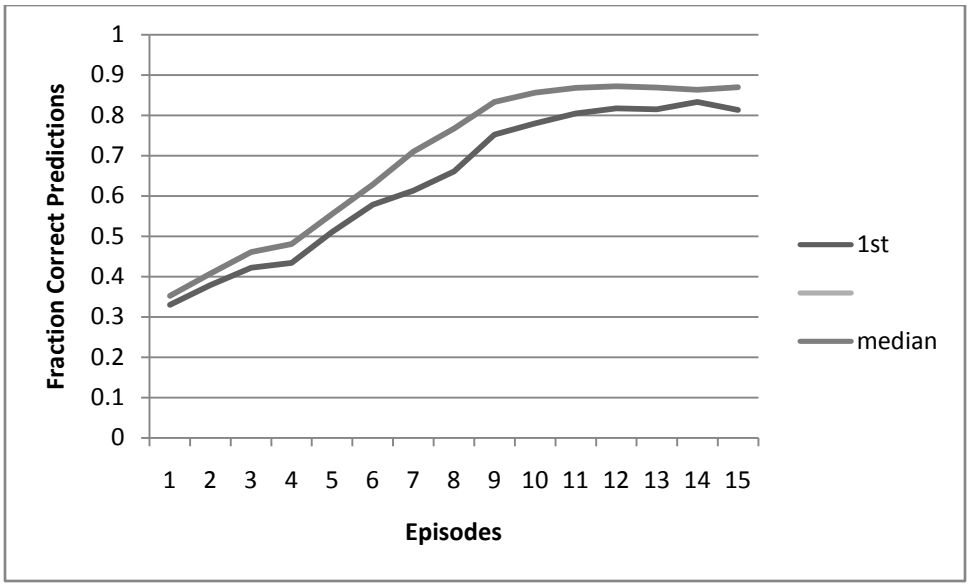


Figure 9.21: Fraction of predictions made that are correct for Dynamic Exploration experiment. Compared to the agent without Dynamic Exploration, this agent learns to be much more accurate.

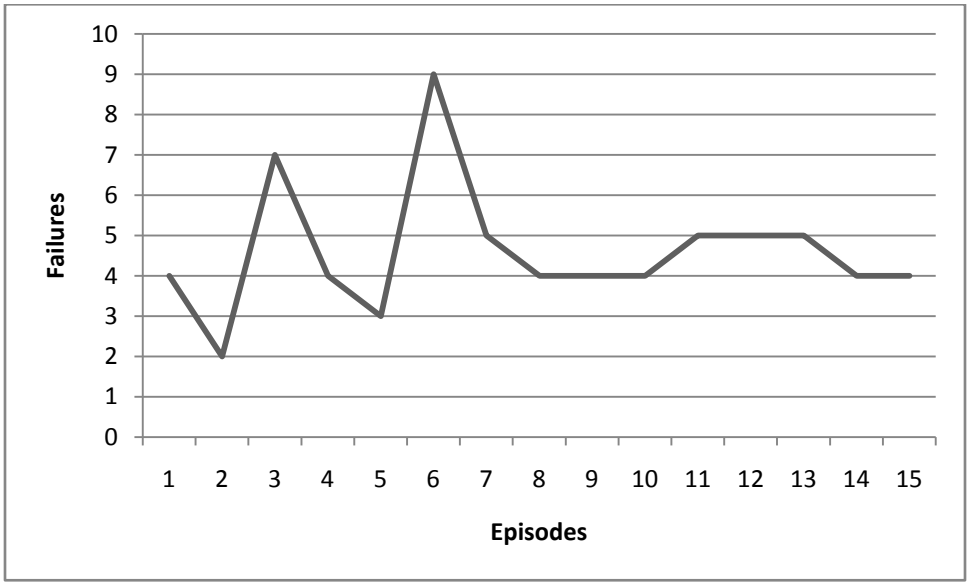


Figure 9.22: Failures across trials for each episode for Dynamic Exploration experiment. Early failures are more erratic but settle at a constant level.

Final Episode Failures	Trial Failures	Total Failures
4	19	69

Table 9.10: Failures for Dynamic Exploration experiment. The agent has more failures than without Dynamic Exploration.

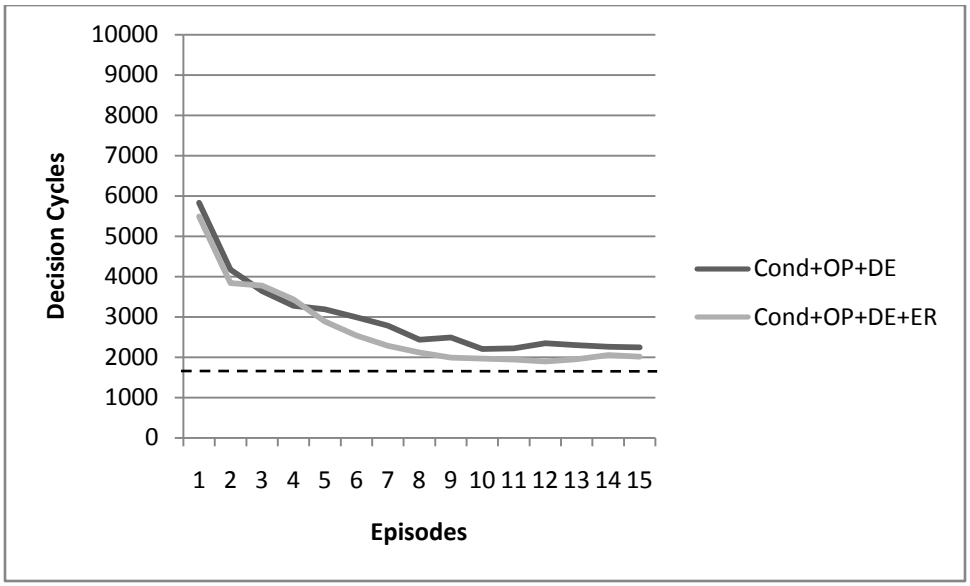


Figure 9.23: Decision cycles to task completion with and without Dynamic Exploration (medians). With Dynamic Exploration, the agent learns slightly faster and better.

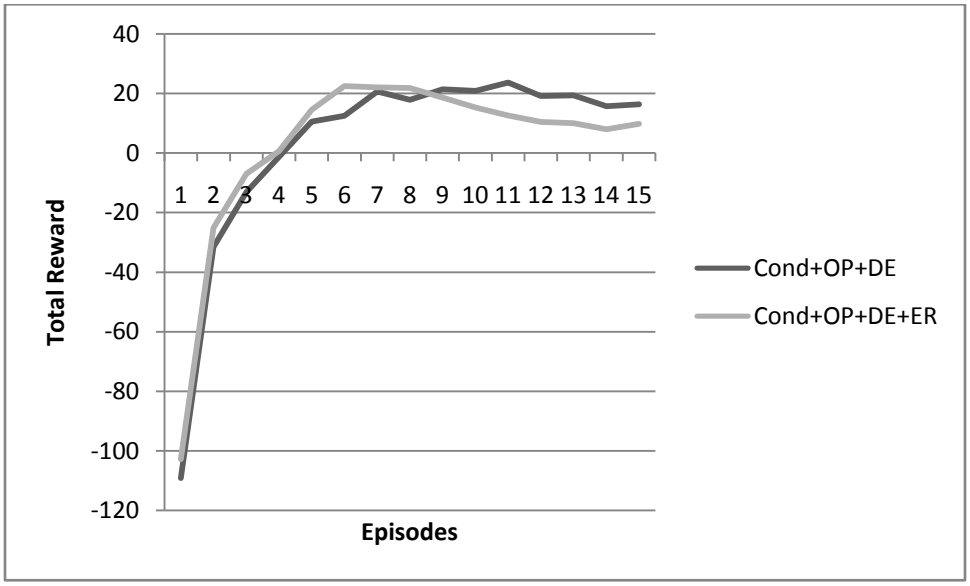


Figure 9.24: Total reward accumulated during task with and without Dynamic Exploration (medians).
With Dynamic Exploration, the reward peaks sooner and reduces faster.

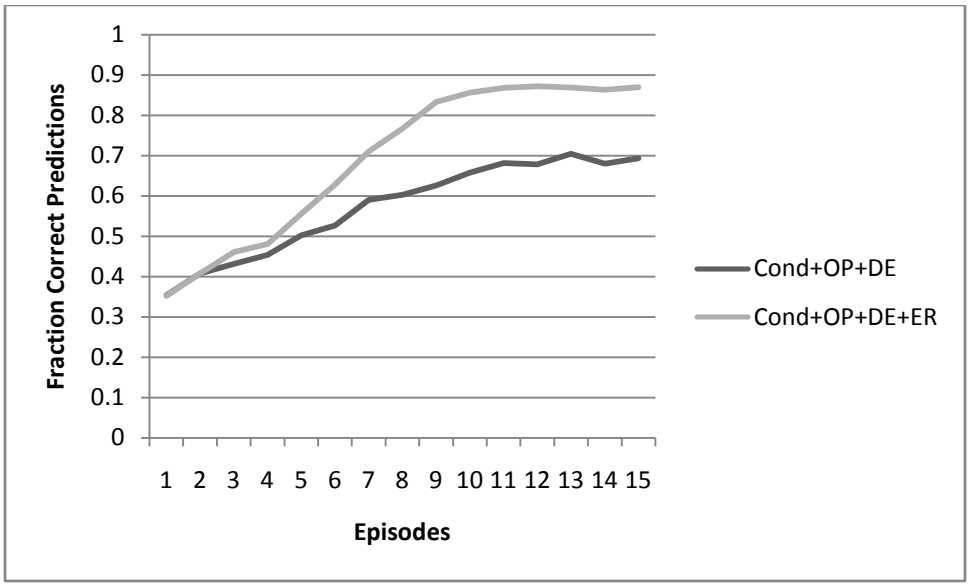


Figure 9.25: Fraction of predictions made that are correct with and without Dynamic Exploration (medians).
The agent with Dynamic Exploration learns much faster and better.

With Dynamic Exploration, the agent's performance generally improved. The change in decision cycles was small, but the agent does seem to converge more quickly (Figure 9.19 and Figure 9.23). The total reward peaked a little earlier, and reduced more (Figure 9.20 and Figure 9.24). This was not surprising given the pronounced improvement in prediction accuracy (about 15% at the end, with a faster initial climb; Figure 9.21 and Figure 9.25). The implication was that exploration rate did in fact reduce over time, and that the agent converged to consistent good behavior.

Where the agent did not improve is in failures (Figure 9.22 and Table 9.10). Whereas before the agent had no failures, it now had many (although still few final failures, implying it usually recovers). The cause for this was probably those cases where the agent had a small negative reward. Still, the fact that failures settle at a constant level implies that the agent's exploration rate lowers and that the agent has settled into a consistent pattern of behavior. The agent could probably do better, but the small magnitude means the exploration rate in these situations is very low. In essence, the agent was locked into this bad behavior for a long time. The only way to break out was either to accumulate enough negative reward to reduce the value sufficiently (which

would take a long time given the small magnitude), or to "get lucky" enough times to find a better option. Still, as the quartiles in the other results show, these failures are outside the norm.

With mood enabled, the agent's performance (not shown) matched the above results. Given that mood had no impact on the agent without Dynamic Exploration, this is not surprising.

9.9.3 Dynamic Learning Rate

As with exploration rate, traditional reinforcement learning systems typically change learning rate in a fixed way, and we found this unsatisfactory for the same reasons as above.

As with Dynamic Exploration Rate, we tried tying the learning rate to the agent's feeling intensity and valence. In situations where the agent has strong feelings, the implication is that there is more to learn, whereas when feelings are weak, there is less to learn. Thus, Dynamic Learning Rate was defined as the absolute value of the feeling intensity multiplied by the feeling valence (that is, the absolute value of the reward).

There may be an interesting interaction between learning rate and episodic memory (via the Outcome Probability and Discrepancy from Expectation appraisals). For example, when the agent first learns about something particularly good, it should learn a lot about it, but as it repeats that action many times, the feeling intensity goes down. However, with Dynamic Learning Rate, the learning rate will go down, too, which will effectively lock in a higher value. Thus, this value will not go down to zero over time, so the agent should never "unlearn" the higher value. Additionally, if the agent ever makes a bad exploratory move, it should get a large negative penalty (since it is unexpected) and that should come with a high learning rate, helping the agent to learn about negatively valued actions quickly.

We performed experiments with Conduciveness, Outcome Probability, and Discrepancy from Expectation enabled. The agent learned across 15 episodes with a 10000 decision cycle cutoff.

In all cases, the results for number of decision cycles, total reward, and fraction correct predictions are virtually identical to the same conditions without Dynamic Learning Rate, and thus these are not shown. Instead, we focus on the failures.

In the first experiment, we simply enabled Dynamic Learning Rate. This matched the original results obtained with a fixed learning rate (Table 9.3); that is, there were no failures.

In the next experiment, we enabled both Dynamic Learning Rate and Dynamic Exploration Rate. This time, the agent performed much better (Figure 9.26 and Table 9.11), than with Dynamic Exploration alone (Figure 9.22 and Table 9.10), but still more than zero. However, the data show that all failures occurred early; once some initial learning took place, agent never failed.

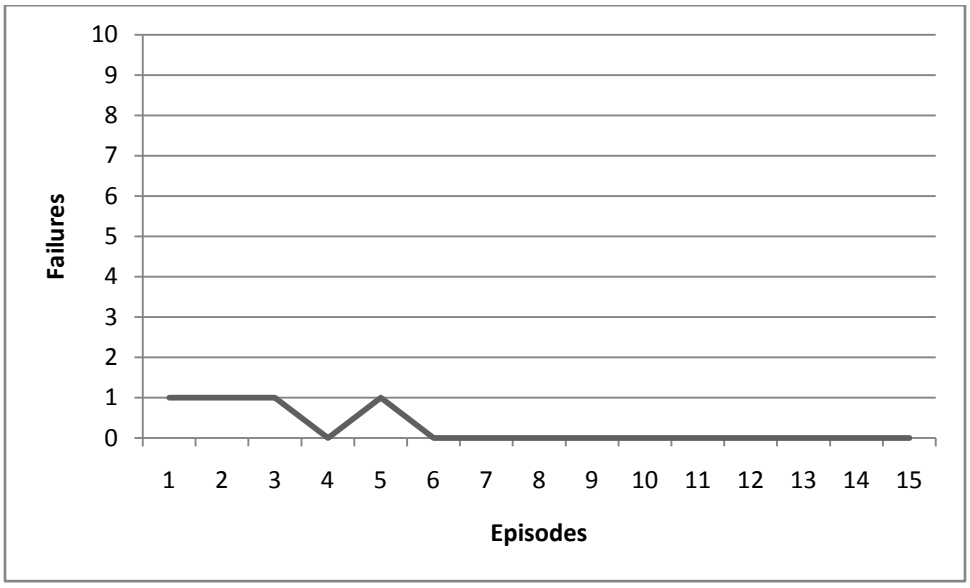


Figure 9.26: Failures across trials for each episode for Dynamic Learning and Exploration experiment.
Some early failures but none in later episodes.

Final Episode Failures	Trial Failures	Total Failures
0	4	4

Table 9.11: Failures for Dynamic Learning and Exploration experiment.
The agent has far fewer failures than with Dynamic Exploration alone.

Finally, we tried Dynamic Learning and Exploration with mood enabled. The agent performed slightly worse under this condition (Figure 9.27 and Table 9.12). This may be variability in the data (since mood has previously had no effect), or it may be that mood, in effect, actually introduces a little bit of noise. The fact that the failures are more spread out across episodes supports the noise explanation, although the agent was still able to recover by the end.

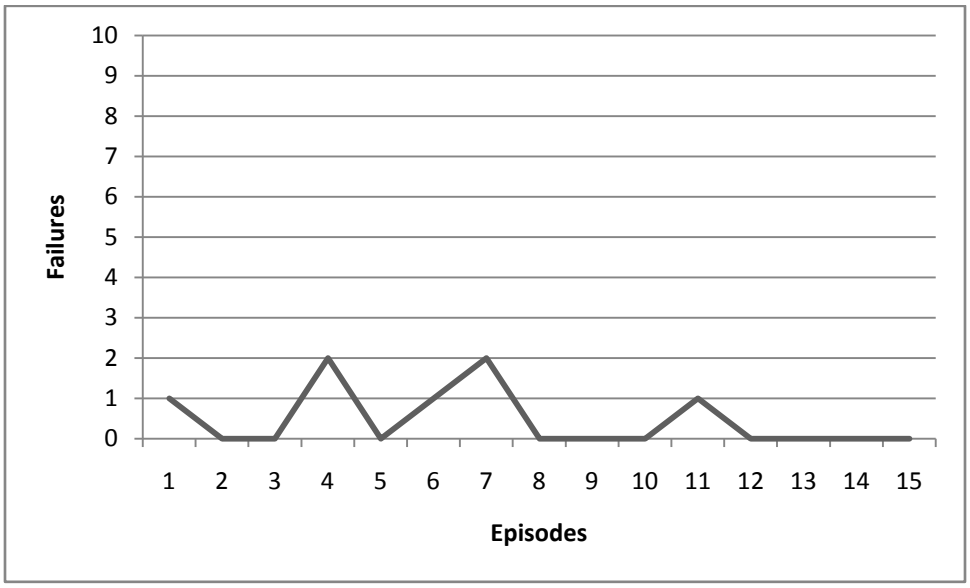


Figure 9.27: Failures across trials for each episode for Dynamic Learning and Exploration experiment with mood.
Failures are more spread out across episodes.

Final Episode Failures	Trial Failures	Total Failures
0	7	7

Table 9.12: Failures for Dynamic Learning and Exploration experiment with mood.
The agent had slightly more failures than without mood.

9.10 Summary

Our goal was twofold. First, we wanted to demonstrate that the system was still capable of learning in a complex, continuous environment. Second, we wanted to explore how various appraisals influenced that learning, in an attempt to tease apart the effects of the different parts of what has become a complex system.

For the first point, the agent was clearly able to learn in the new domain. This was repeatedly demonstrated in our experiments used to address the second point. For the second point, we chose a subset of five appraisals (the rest were disabled). We tested Conduciveness alone, which was generally successful but suffered from some failures. Next, we introduced Outcome Probability and Discrepancy from expectation. The values for these were tied to the system's prediction ability, which was enhanced with an episodic memory. This resulted in zero failures. This setup with Conduciveness, Outcome Probability, Discrepancy from Expectation, and the episodic memory was used as a base for the remaining experiments.

To this base we first added Goal Relevance via a boosting mechanism, which almost did too well. We argued that this did not obviate the need for the other appraisals, and reinforced this with a second experiment with reduced knowledge, which demonstrated that learning still takes place. Next we added Intrinsic Pleasantness to the base. This took the form of positive valence for blocks. As expected, the results were mixed, but overall the agent did well and learned to recover from negative situations. We also explored mood in the context of the base model, but found that it had no impact. Finally, we tried dynamically changing the exploration and learning rates. Dynamic Exploration rate had a generally positive effect, but resulted in more failures. By itself, Dynamic Learning Rate had no effect, but when combined with Dynamic Exploration Rate, it retained the benefits of Dynamic Exploration while also reducing failures. An important takeaway from these dynamics experiments is that emotion was able to regulate another part of the cognitive architecture with positive results.

Table 9.13 summarizes the key points for each of the experiments we ran.

Conduciveness	Foundation to learning. Agent learns to perform the task better over time.
Outcome Probability and Discrepancy from Expectation	Introduced episodic memory for generating predictions as basis for generating values for these appraisals. Agent learns to predict better over time. Also results in much improved failure rates.
Goal Relevance	Used to "boost" value of proposed Attend operators. Agent does extremely well (except for failures), to the point where it almost isn't learning, raising questions about the value of other appraisals. Knowledge about Goal Relevance was reduced, leading to more learning.
Intrinsic Pleasantness	Used to provide a task-independent bias on valence. Results are mixed, as expected, but agent generally learns to overcome problems.
Mood	Mood had no effect in this domain under a variety of circumstances.
Dynamic Exploration and Learning Rates	Emotion was used to regulate another part of the cognitive architecture. Dynamic Exploration Rate resulted in tighter convergence and better prediction accuracy, but more failures. Dynamic Learning Rate had no impact alone (e.g., failures were still zero), but combined with Dynamic Exploration, retained the benefits of Dynamic Exploration while reducing failures.

Table 9.13: Summary of experiments and key takeaway points from the results.

Finally, we want to summarize our contributions to reinforcement learning. We connected emotion, mood and feeling to reinforcement learning by hypothesizing the use of feeling as an intrinsically motivating reward signal (Chapter 7). In this context, mood provides a time-averaging effect over reward, allowing rewards to be generated in states that lack external stimuli. This method of generating reward worked in both discrete and continuous environments (Chapters 7-9). In the continuous case, we introduced a method for skipping over temporal gaps between RL states as caused by impasses and non-learned processing (section 8.3.4). Finally, we demonstrated how aspects of the emotion system can be used to manipulate aspects of the reinforcement learning system; specifically, we used emotion to regulate the learning and exploration rate parameters.

Chapter 10

Summary, Future Work, and Conclusion

In summary, we have presented an integration of cognition and emotion, in which each side provides a functional necessity to the other: cognition (as PEACTIDM) provides the processes necessary to generate emotions (via appraisals), whereas emotion (via appraisals) provides the data which cognition (via PEACTIDM) processes. On top of this theoretical grounding, we extended the Soar cognitive architecture to include the computational mechanisms necessary to support our proposed integration.

This integration was realized in a Soar model of the simple choice response task (Chapter 3). This model demonstrated that cognition and emotion both provided insights about each other, including new mechanisms for cognitive architectures and representations for emotions, including active appraisal frames, the appraisal detector, and emotion intensity. We also touched on human data.

To explore further, we then extended the model to an ongoing task in the artificial Eaters environment (Chapter 4). For this domain, we expanded the set of appraisals the model generated. The new domain also introduced many new issues, such as creation of subtasks and long-term influences of emotion. The later led us to propose a model of the interaction between emotion, mood and feeling. We also demonstrated a connection between emotion and behavior (in both directions), including the temporal dynamics, in our evaluation of this model (Chapter 5).

Work in this non-learning model begged the question: what are the functional benefits of emotion? In response to this question, we proposed that it may serve as an intrinsic motivator (that is, internal reward signal) for reinforcement learning (Chapter 7). Our verification of this began with an extension of the non-learning model so that it could

learn in the same simple Eaters domain. This was successful: the agent was able to improve its performance in the domain. However, the complexity of the system made it difficult to tell how various aspects of the model actually contributed to its performance. Additionally, we wanted to explore how the model would scale to a more complex domain.

To explore these issues, we introduced the Room domain, which has continuous time and space properties. The model was revised in several ways to deal with this new domain, including changes in knowledge and refinements to the way various aspects worked (Chapter 8). To explore performance in this new environment (Chapter 9), we enabled only a subset of appraisals in various configurations. We also introduced a new episodic memory mechanism to better support the generation of predictions and the associated Outcome Probability and Discrepancy from Expectation appraisals. We demonstrated that each of the selected appraisals influences the agent's learning and behavior, usually positively. We also explored mood in this domain, but with negative results. Finally, we looked at using feelings to dynamically modulate exploration and learning rates. This resulted is generally improved performance.

There are vast, overlapping areas we have yet to explore. One goal is to expand to a more complete model of emotion, including its integration with the rest of cognition and physiology. This expansion will likely provide additional constraints to help shape our theory, and our theory may provide additional constraints on the theories in these areas. For example, how we represent appraisals and emotion may be influenced by these other areas, and vice versa. Besides these areas, we will also discuss scalability, and validation.

Beyond our very abstract mood model, the system has no notion of physiology. Physiology plays critical roles in action tendencies (Frijda et al., 1989), non-verbal communication such as facial expression (Ekman, et al., 1987) and tone of voice, and other more basic physiological measures such as skin conductance, heart rate, and blood pressure. Once a more complete physiological model is in place, we can also explore introspection about the current physiological state, for example, which may extend to

emotion recognition (Picard, 1997). Basic drives such as hunger and thirst can also be explored in the context of emotion.

On the cognitive side, we have scratched the surface of learning, but that remains a major area for continued research. For example, we have not yet explored how appraisal values might be learned. We also need to explore how emotion interacts with other cognitive mechanisms; for example, the episodic and semantic memories depicted in Figure 2.2. Such a system could allow phenomena ranging from priming effects (Neumann, 2001) to emotional intelligence (Picard, 1997) to be explored. While we did scratch the surface of actually allowing emotion to influence decision making, clearly there is much more to be done there as well.

We demonstrated that the system can scale to a more complex environment with more complex appraisal value generation. But there is also the matter of simply generating more appraisals; for example, what about socially oriented appraisals? Does the system scale to explaining aspects of social interaction and culture?

Finally, there is the issue of validation. There are multiple ways in which we might attempt to validate the system going forward: believability (Neal Reilly, 1996), human data (which we explored briefly), physiological measures, behavior, and decision making (Gratch et al., 2006), and functionality (e.g., learning, which we have already started to explore, the impact of additional appraisals, etc.).

This partial list demonstrates the vast amount of work remaining; it seems unlikely that anything short of a complete human intelligence system can actually address it all. Indeed, this is perhaps a key point that emotion researchers have been making for a long time: emotion is a key aspect of human-level intelligence.

In conclusion, our system has several features and implications, which we list below. For ease of understanding, we have included the list we presented earlier (section 5.3) and extended it here:

1. Appraisals are a functionally required part of cognitive processing; they cannot be replaced by some other emotion generation theory.

2. Appraisals provide a task-independent language for control knowledge, although their values can be determined by task-dependent knowledge. Emotion and mood, by virtue of being derived from appraisals, abstract summaries of the current and past states, respectively. Feeling, then, augments the current state representation with knowledge that combines the emotion and mood representations and can influence control.
3. The integration of appraisal and PEACTIDM implies a partial ordering of appraisal generation.
4. This partial ordering specifies a time course of appraisal generation, which leads to time courses for emotion, mood and feeling.
5. Emotion intensity is largely determined by expectations and consequences for the agent; thus, even seemingly mundane tasks can be emotional under the right circumstances.
6. In general, appraisals may require an arbitrary amount of inference to be generated. That is, the theory supports Marsella & Gratch's (in press) distinction between appraisal and inference.

The following are additions to the original list: Internal and external stimuli are treated identically.

8. Circumplex models can be synthesized from appraisal models: they provide a description of the emotion generated by appraisal.
9. Reinforcement learning is driven by intrinsically generated rewards based on the agent's feeling.
10. Mood averages reward over time, allowing states with no reward-invoking stimulus to still have a reward associated with them. This leads to improved learning in some cases.
11. The system scales to continuous time and space environments. Changes made to support this include adding a temporal gap jump to reinforcement learning to allow rewards to propagate back through non-RL states.
12. Each appraisal contributes to the agent's performance.
13. Reinforcement learning parameters can be influenced by the current emotional state, resulting in improved performance.

Emotion has become an active area of research in recent years. We hope this thesis makes a meaningful contribution to this area, and reminds researchers in both the cognitive architecture and emotion camps of the important role each must play in the search for understanding.

References

Agre, P. (1988). *The dynamic structure of everyday life.* Dissertation, MIT, Electrical Engineering and Computer Science, Cambridge.

Anderson, J. R. (2007). *How Can the Human Mind Exist in the Physical Universe?* New York: Oxford University Press.

Breazeal, C. (2003). Function Meets Style: Insights from Emotion Theory Applied to HRI. *IEEE Transactions in Systems, Man, and Cybernetics, Part C , 34* (2), 187-194.

Chong, R. S., & Laird, J. (1997). Identifying Dual-Task Executive Process Knowledge using EPIC-Soar. *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Damasio, A. (1994). *Descarte's error: Emotion, reason, and the human brain.* New York: Avon Books.

Damasio, A. (2003). *Looking for Spinoza: Joy, sorrow, and the feeling brain.* USA: Harcourt.

Diener, E., & Diener, C. (1996). Most people are happy. *Psychological Science , 7* (3), 181-185.

Ekman, P., Friesen, W., O'Sullivan, M., Chan, A., Diacoyanni-Tarlatzis, I., Heider, K., et al. (1987). Universals and Cultural Differences in the Judgments of Facial Expressions of Emotion. *Journal of Personality and Social Psychology , 53* (4), 712-717.

Feldman Barrett, L., & Russell, J. (1998). Interdependence and bipolarity in the structure of current affect. *Journal of Personality and Social Psychology , 74*, 967-984.

Forgas, J. P. (1999). Network theories and beyond. In T. Dalgleish, & M. Power (Eds.), *Handbook of Cognition and Emotion* (pp. 591-611). Chichester, England: Wiley and Sons.

Fredrickson, B., & Levenson, R. (1998). Positive emotions speed recovery from cardiovascular sequelae of negative emotions. *Cognition and Emotion , 12* (2), 191-220.

Frijda, N. H., Kuipers, P., & ter Schure, E. (1989). Relations among emotion, appraisal, and emotional action readiness. *Journal of Personality and Social Psychology , 57*, 212-228.

Gratch, J., & Marsella, S. (2004). A domain-independent framework for modeling emotion. *Cognitive Systems Research , 5* (4), 269-306.

Gross, J. J., & John, O. P. (2003). Individual differences in two emotion regulation processes: Implications for affect, relationships, and well-being. *Journal of Personality and Social Psychology* (85), 348-362.

Hogewoning, E. (2007). *Strategies for Affect-Controlled Action-Selection in Soar-RL*. Technical Report 07-01, Leiden University, Leiden Institute of Advanced Computer Science.

Hogewoning, E., Broekens, J., Eggermont, J., & Bovenkamp, E. (2007). Strategies for Affect-Controlled Action-Selection in Soar-RL. In J. Á. Mira (Ed.), *Nature Inspired Problem-Solving Methods in Knowledge Engineering, Second International Work-Conference on the Interplay Between Natural and Artificial Computation. 2*, pp. 501-510. La Manga del Mar Menor, Spain: Springer.

Hudlicka, E. (2004). Beyond Cognition: Modeling Emotion in Cognitive Architectures. *Proceedings of the International Conference of Cognitive Modelling, ICCM 2004*, (pp. 118-123). Pittsburgh, PA.

John, B. E., Rosenbloom, P. S., & Newell, A. (1985). A theory of stimulus-response compatibility applied to human-computer interaction. *Proceedings of CHI'85 Human Factors in Computer Systems.* New York: Association for Computing Machinery.

Kaplan, S. S., & Chown, E. (1991). Tracing Recurrent Activity in Cognitive Elements (TRACE): A Model of Temporal Dynamics in a Cell Assembly. *Connection Science , 3*, 179-206.

Kaplan, S., & Kaplan, R. (1982). *Cognition and Environment.* New York: Praeger.

Kieras, D., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction , 12*, 391-438.

Laird, J. (2008). Extending the Soar Cognitive Architecture. *Proceedings of the First Conference on Artificial General Intelligence.* Memphis: IOS Press.

Laird, J., Rosenbloom, P., & Newell, A. (1986). Chunking in Soar: The Anatomy of a General Learning Mechanism. *Machine Learning , 1* (1), 11-46.

Lambie, J. A., & Marcel, A. J. (2002). Consciousness and the varieties of emotion experience: A theoretical framework. *Psychological Review , 109* (2), 219-259.

Lathrop, S. (2008). *Extending Cognitive Architectures with Spatial and Visual Imagery Mechanisms.* Dissertation, University of Michigan, Electrical Engineering and Computer Science, Ann Arbor.

Marsella, M., & Gratch, J. (in press). EMA: A Process Model of Appraisal Dynamics. *Journal of Cognitive Systems Research* .

Nason, S., & Laird, J. (2005). Soar-RL, Integrating Reinforcement Learning with Soar. *Cognitive Systems Research , 6* (1), 51-59.

Neal Reilly, W. S. (1996). *Believable Social and Emotional Agents.* Technical Report CMU-CS-96-138, Carnegie Mellon University, Pittsburgh.

Neal Reilly, W. S. (2006). Modeling what happens between emotional antecedents and emotional consequents. *Proceedings of the Eighteenth European Meeting on Cybernetics and Systems Research* (pp. 607-612). Vienna, Austria: Austrian Society for Cybernetic Studies.

Neumann, R. (2001). The causal influences of attributions on emotioins: A procedural priming approach. *Psychological Science , 11* (3), 179-182.

Newell, A. (1990). *Unified Theories of Cognition.* Cambridge: Harvard University Press.

Newell, A., Shaw, J. C., & Simon, H. A. (1960). Report on a general problem solving program. *Proceedings of the International Conference on Information Processing* (pp. 256-264). Paris: UNESCO.

Nuxoll, A. (2007). *Enhancing Intelligent Agents with Episodic Memory.* Dissertation, University of Michigan, Electrical Engineering and Computer Science, Ann Arbor.

Ortony, A., Clore, G., & Collins, A. (1988). *The Cognitive Structure of Emotions.* Cambridge, MA: Cambridge University Press.

Picard, R. (1997). *Affective Computing.* Cambridge, MA: MIT Press.

Roseman, I., & Smith, C. A. (2001). Appraisal Theory: Overview, Assumptions, Varieties. In K. Scherer, A. Schorr, & T. Johnstone (Eds.), *Appraisal Processes in Emotion: Theory, Methods, Research.* New York: Oxford University Press.

Rosenberg, E. L. (1998). Levels of analysis and the organization of affect. *Review of General Psychology , 2*, 247-270.

Salichs, M., & Malfaz, M. (2006). Using Emotions on Autonomous Agents. The Role of Happiness, Sadness, and Fear. *Proceedings of the AISB'06 Adaptation in Artificial and Biological Systems*, (pp. 157-164). Bristol, UK.

Scherer, K. (2001). Appraisal Considered as a Process of Multilevel Sequuential Checking. In K. Scherer, A. Schorr, & T. Johnstone (Eds.), *Appraisal Processes in Emotion: Theory, Methods, Research.* New York: Oxford University Press.

Schorr, A. (2001). Appraisal: The Evolution of an Idea. In K. Scherer, A. Schorr, & T. Johnstone (Eds.), *Appraisal Processes in Emotion: Theory, Methods, Research* (pp. 20-34). New York: Oxford University Press.

Singh, S., Barto, A., & Chentanez, N. (2004). Intrinsically Motivated Reinforcement Learning. *Proceedings of Advances in Neural Information Processing Systems 17 (NIPS).*

Smith, C. A., & Kirby, L. A. (2001). Toward delivering on the promise of appraisal theory. In K. Scherer, A. Schorr, & T. Johnstone (Eds.), *Appraisal Processes in Emotion: Theory, Methods, Research* (pp. 121-138). New York: Oxford University Press.

Smith, C. A., & Lazarus, R. S. (1990). Emotion and adaptation. In L. A. Pervin (Ed.), *Handbook of Personality Theory and Research* (pp. 609-637). New York: Guilford.

Sun, R. (2006). The CLARION cognitive architecture: Extending cognitive modeling to social simulation. In R. Sun (Ed.), *Cognition and Multi-Agent Interaction.* New York: Cambridge University Press.

Sutton, R., & Barto, A. (1998). *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press.

Wang, Y., & Laird, J. (2006). *Integrating Semantic Memory into a Cognitive Architecture.* Technical Report CCA-TR-2006-02, University of Michigan.

Wilson, S. (1996). Explore/Exploit Strategies in Autonomy. In P. Maes, M. Mataric, J. Pollack, J. Meyer, & S. Wilson (Ed.), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior.* Cambridge, MA: MIT Press/Bradford Books.

Yik, M., Russell, J., & Feldman Barrett, L. (1999). Structure of Self-Reported Current Affect: Integration and Beyond. *Journal of Personality and Social Psychology , 77* (3), 600-619.

Zajonc, R. (1984). On the primacy of affect. *American Psychologist , 39*, 117-123.