Wesleyan University

Demonstrations of Dynamical Intention for Hybrid Agents

by

Henny Admoni
Faculty Advisor: Dr. Eric Aaron

A Dissertation submitted to the faculty of Wesleyan University
in partial fulfillment of the requirements for the degree of
Master of Arts

Middletown, Connecticut April, 2009

Acknowledgments

This thesis would not exist without collaboration, support and guidance from my research advisor, Professor Eric Aaron. Hybrid dynamical cognitive agents are his idea; my original contributions to the project are ideas about learning for HDCAs in Chapter 4, and MATLAB implementations of HDCA cognition, including demonstrations in Chapter 5. Thank you, Professor Aaron, for letting me run with your ideas and for helping me steer a good path. My committee members, Professors Daniel Krizanc and Michael Rice, were very helpful in the construction of this document; their comments and questions illuminated ways to improve my presentation of HDCAs. Thank you for your hard work in reading multiple versions of my thesis, and for being the nicest committee members I could have asked for at my defense. Many of my peers helped me develop this thesis: Some provided me with information from their scientific fields; others listened to presentations of my work and asked helpful questions. It is inspiring to have a network of scientific peers to turn to, and I thank you all. As an undergraduate, I completed a self-designed major, and the committee for that major deserves much credit for allowing me to explore my academic passions. Support from Professors Eric Aaron, Andrea Patalano and John Kirn helped me channel my interests in computer science, cognitive psychology, and neuroscience into this thesis work. Part of the work for this thesis was completed with generous support from the Hughes Summer Research Program. Additionally, the Mathematics and Computer Science Department has been my home base for most of my time at Wesleyan University. Both deserve thanks for allowing me to develop as a scholar. Through it all, my family has been the loudest cheering squad, and home has been the place I go to step back from research for a while. Thank you Ima, Aba, Sha-har and Netta for your love and support this year and always.

Contents

List of Figures

List of Tables

Abstract

Representations of intention shared by reactive and deliberative systems of hybrid agents enable seamless integration of high-level logical reasoning and low-level behavioral response. This thesis presents an architecture for *hybrid dynamical cognitive agents* (*HDCAs*), hybrid reactive/deliberative agents with cognitive systems of continuously evolving *beliefs*, *desires*, and *intentions* based on *BDI* and *spreading activation network* models. *Dynamical intentions* support goal-directed behavior in both reactive and deliberative systems of HDCAs: on the reactive level, dynamical intentions allow for continuous cognitive evolution and real-time task re-sequencing; on the deliberative level, dynamical intentions enable logical reasoning and plan generation. Because intention representations are shared between both systems, reactive behavior and goal-directed deliberation are straightforwardly integrated in HDCAs. Additionally, *Hebbian learning* on connections in the spreading activation network of beliefs, desires, and intentions trains HDCAs' reactive systems to respond to typically deliberative-level information. To establish comparability between HDCAs and traditional BDI-based architectures, dynamical intentions are shown to be consistent with the philosophical definition of intention from other BDI models. Simulations of autonomous, embodied HDCAs navigating to complete tasks in a grid city environment illustrate dynamical, intention-based behavior that derives from clean integration of reactive and deliberative systems.

CHAPTER 1

Introduction

Intelligent autonomous agents that perform *goal-directed behavior*—action to achieve desired results—require some representation of goals and commitments. One implementation for goal-directed behavior is called the *Belief-Desire-Intention* (*BDI*) model; in this framework, *intentions* define agents' commitments to action, which determine which actions are ultimately performed. In BDI models, agents reason about current intentions to form *plans*, sequences of desired behaviors or states of the environment. This thesis introduces a new agent framework that extends traditional BDI models with *dynamical intentions*, continuously evolving commitments to action; agents in this new framework, called *hybrid dynamical cognitive agents* (*HDCAs*), have actions that are tightly coupled to their dynamical intentions.

As an introduction to HDCAs, imagine two embodied, autonomous agents navigating to locations in a grid city in order to complete errands, such as depositing a check at the bank, `DepositCheck`, and mailing a letter at the post office, `MailLetter`. One agent, A_D, is a typical *deliberative* BDI agent; A_D logically reasons about which actions might optimize its performance, using symbolic representations of the world (including of its own intentions) to construct an internal world model [**14**]. The second agent, A_R, is a *reactive* agent that maintains no symbolic representations; instead, A_R behaves reflexively in response to the state of its *dynamical intentions*, commitments to tasks which have continuously varying *activation* values indicating how relevant or salient these commitments are to the agent at any given time. High activation values on intentions mean that those intentions are important in A_R's cognitive system; in fact, the most activated (or *maximal*) intention specifies which action A_R is currently performing. In addition to intentions, both agents have *beliefs*—knowledge about the

world—and *desires*—actions the agents wish to achieve. As with intentions, A_D represents beliefs and desires as propositional statements, while A_R's beliefs and desires have activation levels indicating salience.

As they navigate to complete their tasks, A_D and A_R use different mechanisms to select actions. A_D first senses its environment and builds a model of the world with symbolic representations, then uses logic to prove that, given the world model and a set of actions with known effects, it can perform a certain sequence of actions to fulfill its goals (i.e., to complete its errands). After generating this action sequence, or *plan*, A_D executes these actions in order; when complete, A_D once again senses the environment and builds a new world model. This *sense-plan-act* cycle underlies traditional deliberative systems [**14**].

In contrast, A_R avoids explicit logical reasoning, instead relying on continuous evolution of beliefs, desires, and (importantly) dynamical intentions to guide action selection. These BDI elements are connected in a *spreading activation network* inspired by (though significantly different from) connectionist architectures such as [**23, 48**]. Related BDI elements share connections over which activation can spread; the amount of activation spreading from one element to another is proportional to the activation of the first element and the strength of the connection. Connections can also be negative, in which case the amount of activation removed from the second element is proportional to the activation of the first. A one-to-one correspondence exists between actions and intentions, and A_R behaves reflexively by selecting the action that is associated with its most highly activated (maximal) intention. When another intention's activation exceeds the current maximal intention's activation, A_R switches to performing the new maximal intention's action. This type of behavior can be seen as a *sense-act* cycle: A_R perceives its environment and allows activation to spread among BDI elements, then behaves according to the current state of those elements, without explicit symbolic reasoning about its behavior.

In some cases, A_R's approach avoids computational effort and can gracefully handle inconsistencies that would pose a problem for A_D. For instance, suppose both agents begin with a conflict in their cognitive systems: They believe that their respective letters have been mailed (the belief, abbreviated B_{LM}, is true or highly activated), but both agents still have positive commitments to mail their letters (intention I_{ML} is also true or highly activated). This kind of inconsistency might be problematic to A_D, which has the commitment to mail a letter but does not believe that the precondition for the action (B_{LM} being false) is true. Simply identifying that an inconsistency is present might be computationally taxing if A_D's knowledge base is large. In contrast, A_R is able to dynamically alter its cognitive system to resolve the inconsistency: As it navigates, continuous negative spreading activation from the conflicting belief B_{LM} draws activation of I_{ML} down until the intention is no longer cognitively salient.

Other circumstances requiring explicit reasoning are better handled by A_D. Suppose both agents begin with intentions to deposit checks at the bank (I_{DC}). As they navigate toward the bank, intentions to pick up their respective children from school (I_{GC}) become more important or salient to both agents. The agents might re-consider their actions at this point: should they continue with the previous task of depositing checks? switch to picking up their children? select completely different tasks? A number of factors might influence this decision, including distances to new task targets (in this case, the school), activations on other intentions, and domain-specific rules and biases (e.g., minimizing total distance traveled; only buying a book or borrowing a book, but not both). For reasoning about factors outside of immediate perception, such as distance information and domain-specific rules, A_D's symbolic world model allows straightforward logical deliberation that A_R's spreading activation network does not.

This thesis presents a framework for *hybrid reactive/deliberative* agents that have deliberative systems, like A_D's, to perform logical reasoning, as well as reactive systems, like A_R's, for real-time dynamical behavior. The next sections in this chapter outline the structure of hybrid dynamical cognitive agents, then provide background in related

areas. Chapter 2 describes technical components of the HDCA framework, and Chapter 3 details how the structure of HDCAs leads to intelligent, realistic agent behavior. Chapter 4 describes how HDCAs learn to use typically deliberative information in their reactive systems. Chapter 5 provides descriptions of demonstrations and experiments conducted to illustrate HDCA behavior, and Chapter 6 includes a summary of the model, a discussion, and suggestions for future work. An appendix provides excerpts of MATLAB code from HDCA implementations.

1.1. Hybrid Dynamical Cognitive Agents

Intentional, goal-directed agent behavior, such as completing tasks in sequence as described above, requires goals and commitments to goals. *Reactive* agents can represent these constructs as *dynamical* cognitive elements; in particular, *dynamical intentions* provide arbitrarily precise representations of evolving, real-time commitments to actions. As indicated by the examples, however, purely reactive behavior may be insufficient in environments that require planning or complex reasoning. In these cases, *deliberative* systems might encode beneficial logical structures with which to reason and plan.

This paper presents a framework for *hybrid dynamical cognitive agents* (*HDCAs*) that have reactive cognitive and physical systems cleanly integrated with (presently simple) deliberative systems. These *hybrid reactive/deliberative* agents are extensions of purely reactive agents described in [**7**]; the addition of deliberation allows for high-level reasoning when necessary. Representations of dynamical intentions are shared by both reactive and deliberative systems of HDCAs, allowing straightforward integration of real-time, arbitrarily precise reactive behavior with deliberative reasoning. Additionally, HDCAs can learn general information about their environments and about domain-specific rules, which may reduce burdens of explicit deliberation.

Purely reactive agents are sensor-driven and essentially respond to stimuli with low-level action, as if by reflex [**55**]. These agents are typically designed for dynamic, real-time environments, where quick responses are important. Unlike deliberative agents,

reactive agents do not rely on modeling the world or on reasoning about such models, which requires that models be correct and environments be predictable or static during deliberation [**14**]. Reactive agents have shortcomings, however, because some high-level problems cannot be solved reflexively with only information sensed at decision making time [**55**]. In such cases, reactive systems can be augmented with deliberative systems, which have representations that allow for memory and long-term planning. Hybrid reactive/deliberative agents are meant to take the best parts of both reactive and deliberative systems by relying on the reactive system for dynamical behavior, such as performing actions, and the deliberative system for more complex reasoning, such as forming plans.

Various implementations exist for goal-directed agent behavior; this thesis focuses on the *Belief-Desire-Intention* (*BDI*) model, from [**50**] and others, as inspired by Bratman's theory of intention [**17**]. In essence, Bratman's theory emphasizes the importance of intentions (commitments to action), rather than desires (goals), in determining agent behavior. The theory describes several properties that distinguish intentions from desires: intentions, not desires, are responsible for agent behavior; intentions, once adopted, restrict the adoption of conflicting intentions, whereas an agent can simultaneously have conflicting desires; and once adopted, intentions tend to persist until completion, unlike desires which may be changed at any time. Intentions for HDCAs meet these three criteria, and are therefore analogous to intentions from other BDI implementations[1] (see section 1.2).

In HDCAs, intentions (as well as beliefs and desires) are represented as continuously evolving cognitive elements with activation levels that indicate the salience "in mind" of each element. Higher activations generally mean that the intention (or belief, or desire) exerts more influence on the agent's cognitive system. For instance, strong

[1]Bratman also identified other distinguishing properties, for instance about the future-directedness of intentions, which require representations that are unavailable in reactive systems; these distinguishing properties are not implemented in HDCAs.

commitments to performing actions can be represented by high intention activations for those actions. As described at the beginning of this chapter, BDI elements are connected in a *spreading activation network* which allows elements to influence other elements' salience by spreading positive or negative activation. Dynamics of BDI elements are described further in section 2.3.

For the present HDCA implementation, intentions are in one-to-one correspondence with actions, and agents perform the action of the maximal (most highly-activated) intention. When this changes—for instance, when one intention's activation value supersedes another's through continuous evolution, or when a task is completed and the intention to perform that task decreases rapidly—agents change their behavior. A *plan* in this framework is a sequence of intentions in decreasing order of activation [**1**]; agents start by fulfilling the first intention in the plan—completing the action with the most salient commitment—then fulfilling the second intention, and so on. Agents *dynamically re-plan* when intentions in the plan change order due to continuous evolution. Agents also *deliberatively re-plan* by changing the order of intentions after explicitly considering a number of factors, such as domain-specific rules and distances to targets, in order to maximize efficiency or minimize distance traveled. Re-planning is discussed more fully in sections 3.1.2 and 3.2.1.

In implementations described by this thesis, HDCAs are modeled as *hybrid dynamical systems*, systems with continuously evolving states, or *modes*; each mode is distinct from other modes, and *transitions* between modes occur instantaneously when continuous states reach certain configurations. For HDCAs, modes are actions, continuous behaviors that are distinct from other actions; continuous states include BDI elements with continuously evolving activations as well as physical elements like position and heading angle. Configurations that prompt transitions between modes are encoded as *guards* specific to each mode. One guard condition, for instance, might dictate that agents must transition out of the current mode if the maximally active intention is no longer the intention to do the action represented by that mode. If guard conditions are

met, the guard is said to be *tripped*, and a transition is taken from current mode to new mode as indicated by a transition function. Hybrid dynamical systems are described further in Chapter 2.

HDCAs learn by changing parts of their underlying hybrid dynamical systems or spreading activation networks. *Hebbian learning* (inspired by [**33**]) alters the spreading activation network of beliefs, desires, and intentions, allowing agents to learn associations between aspects of their environment. For instance, agents might use Hebbian learning for traditional classical conditioning: an unconditioned response (e.g., the response of a child to seeing its mother) might be elicited from a conditioned stimulus (e.g., seeing the post office where the child's mother works) simply through repeated pairings of conditioned stimulus and unconditioned response. In this example, the child might begin to run toward the post office expecting its mother even before seeing her, because of previous iterated exposure to its mother at the post office. In a similar way, Hebbian learning can also be used to strengthen connections between certain actions: An agent that knows that the bank is proximate to the bookstore is more likely to commit to going to the bank when it also has an errand to run at the bookstore. Learning for HDCAs is described in Chapter 4.

1.2. Related Work

The design of hybrid dynamical cognitive agents touches upon many fields, including intelligent agents, control theory, machine learning and cognitive science. This section provides some background about these fields as they relate to HDCAs.

1.2.1. Intelligent Agent Architectures. In this thesis, agents are autonomous, embodied, intelligent entities capable of perceiving their environments and acting upon those environments in pursuit of goals. HDCAs are part of a larger class of agents called *hybrid reactive/deliberative* agents; architectures in this class merge *reactive* and *deliberative* systems.

In reactive architectures, complex behavior emerges from many direct, often low-level responses to stimuli. Reactive systems tightly couple agents' perceptions with behaviors using sets of *sense-act* rules that determine particular behaviors in response to specific environmental stimuli. These systems avoid error-prone, time-consuming construction of world models and explicit abstract representations of knowledge that may become invalid as the world changes. Because reactive agents do not logically reason about actions, they do not require that actions have guaranteed effects. These agents perform well in *stochastic*, *nondeterministic* environments, that is, environments that may change unexpectedly and where behavior is not guaranteed to produce certain results [**55**]. Reactive systems are often biologically inspired; according to Rodney Brooks:

> Real biological systems are not rational agents that take inputs, compute logically, and produce outputs. They are a mess of many mechanisms working in various ways, out of which emerges the behavior that we observe and rationalize [**20**, pg. 14].

This viewpoint is echoed in cognitive dynamic systems theory, discussed in section 1.2.6.

One reactive system, the subsumption architecture [**18**], uses layers of control made up of fairly simple computational machines that fulfill individual goals concurrently. Another reactive system called AuRA uses schema theory, which provides behavioral primitives, called schemas, that can be applied in a distributed manner to construct more complex behaviors [**14**]. A third system called Pengi is capable of complex, apparently planned activity without explicit representations of its environment [**8**].

Reactive architectures came about in response to limitations of *deliberative* architectures, traditional top-down approaches for building intelligent agents, which were criticized for their dependence on symbolic representation and their disconnect from biological examples of intelligence [**19**]. Deliberative systems are based on a *sense-plan-act* cycle and explicit symbolic knowledge representation, often in the form of propositional

or first-order logic; reasoning about behavior involves manipulating these logical representations in proofs. Symbolic knowledge representation allows for memory about information that is not immediately perceptible, enabling long-term planning. One canonical deliberative system called STRIPS [**28**], famously implemented on Shakey the Robot [**44**], represents the world as a collection of first-order predicate calculus formulas and uses a theorem prover to plan goal-directed action.

Hybrid reactive/deliberative architectures combine reactive and deliberative systems in an attempt to benefit from their complementary strengths. Hybrid architectures often use deliberation sparingly to adjust reactive behaviors; for example, INTERRAP [**43**] has reactive layers to handle situations requiring quick responses, and deliberative layers to perform more complex goal-directed planning. The Procedural Reasoning System [**30**] uses explicitly represented beliefs, desires, and intentions to create partially-elaborated plans that are executed by a continuously reactive robot. These beliefs, desires, and intentions from PRS are also important in the HDCA architecture.

1.2.2. Beliefs, Desires, Intentions. Reasoning in resource-bounded, dynamic conditions—as is the case with real-world systems—requires a balance of time spent deliberating and time spent acting [**51**]. The Belief-Desire-Intention model is one framework for goal-directed decision making and planning, based on Bratman's philosophical theory of human intention [**17**] and first implemented in Georgeff and Lansky's Procedural Reasoning System [**30**]. A formalized logic has been developed for intentions [**22**]; BDI applications include air-traffic control [**52**], space shuttle design, and telecommunications [**37**].

BDI is well-suited to modeling human behavior because the framework fits with a folk psychological view that people use to describe their own actions [**46**]. This approach has been applied to a variety of human behavioral models such as models of multi-agent systems [**25, 26, 57**], identification of student emotions in intelligent learning environments [**38**], recognition other pilots' plans in air-combat simulations [**34**], and psychologically-based situation recognition [**45**].

Although traditionally applied in deliberative frameworks with explicit knowledge representation, BDI has been implemented in *hybrid reactive/deliberative* architectures as well. For example, systems for coordinating multi-agent behavior [**67**] can dynamically monitor plans using BDI and deliberatively generate new plans using a separate planner. This thesis presents a novel approach that, unlike other BDI systems, fully integrates beliefs, desires and intentions into agents' reactive systems.

1.2.3. Navigation. Navigation is an important and widely-studied aspect of embodied agent behavior. Reactive systems for navigation produce robust behavior in complex, dynamic environments [**13**]. Brooks's subsumption architecture, for instance, has been implemented in robots that wander an office environment [**18**]. Reactive navigation can be coupled with *a priori* [**13**] or acquired [**41**] knowledge to enhance navigation behavior. Groups of individual agents with high-level mental states and reactive behavior control can form complex and realistic crowds [**58**].

Low-level navigation, such as obstacle avoidance, and high-level navigation, such as path selection, have been modeled in animated agents with a hybrid dynamical systems framework [**5, 6**]. This framework is further implemented and extended for HDCAs in this thesis.

1.2.4. Hybrid Dynamical Systems. A *hybrid dynamical system* is a framework for modeling systems with both continuous and discrete parts (see Chapter 2 for a detailed description of hybrid dynamical systems). The combination of continuous and discrete dynamics makes hybrid dynamical systems a natural framework with which to model real-world phenomena such as biomolecular networks [**9**], manufacturing [**47**], and air traffic management [**65**]. A major benefit of hybrid dynamical systems is that such models can be verified: Algorithmic analysis allows researchers to prove that a given state can be reached, to mechanically verify whether safety requirements are met, and to automatically determine constraints on system parameters to guarantee certain behaviors [**10, 11, 36**]. For HDCAs, verification could guarantee that certain agent behaviors will or will not occur under given conditions.

Hybrid dynamical systems have been implemented as models of agent intelligence, in which low-level reactive intelligence is continuous, while high-level behavioral patterns are discrete [**5, 6, 27**]. Similar models have been applied to sophisticated cognitive tasks: a robot observing a human performing a task can construct a plan for the same task using a hybrid dynamical system model [**21**]; obstacle avoidance during navigation by robots [**27**], autonomous vehicles [**68**] and animated agents [**5, 6**] can be implemented with a hybrid dynamical systems framework for agent behavior; and fine-grained facial expressions can be identified using a hybrid dynamical system to extract and analyze dynamic features of a face [**39**].

1.2.5. Connectionist Models. The HDCAs described in this paper use a spreading activation network that resembles some traditional connectionist or neural network architectures. Modern neural network models began with Rosenblatt's "perceptron" [**53**]. Quillian's model of semantic memory [**49**], one of the first connectionist models, has a network of nodes representing concepts interconnected with different kinds of associative links; semantic identification involves spreading of *activation* from input nodes to linked nodes, and activated concepts can be used to infer relationships between words. Quillian's model was later extended and verified experimentally by Collins and Loftus [**23**], who found that the model could explain human behavior on a number of categorization tasks.

Since then, spreading activation has been successfully applied in a variety of connectionist architectures. ACT theory [**12**], built upon a network of nodes with all-or-nothing activation, successfully predicts human memory phenomena like judgments of associative relatedness and effects of extensive practice on memory. Neural networks, such as parallel distributed processing models [**42, 54**], established connectionist models of cognition as challenges to traditional models of symbolic computation, arguing that cognition could be described without formulation of explicit rules. More recent research using connectionist models has included knowledge representation and rapid

inference [**59**], models of goal-directed action selection in situated agents [**40**], and control and learning of object recognition in autonomous robots [**15**].

The work described by this paper uses a spreading activation network inspired by connectionist architectures for the dynamics of agent cognition. In traditional connectionist architectures, knowledge is distributed across multiple nodes; in the current model, individual nodes represent coherent concepts like beliefs about whether tasks have been completed.

1.2.6. Dynamic Systems in Cognitive Science. Models of human cognition seek to formalize cognitive processes such as reasoning, decision making, language acquisition, and problem solving. Traditional approaches to human cognition (e.g., [**29**]) involve processes that manipulate discrete symbolic representations of events in the world. In contrast, cognitive *dynamic systems* theories assert that cognition is an emergent phenomenon resulting from interactions among simple, low-level, non-symbolic processes [**60**]. According to this theory, apparent order in the process of human development often highlighted by other developmental theories is simply the mind as a dynamic system reaching certain stable configurations [**63, 64**]. The HDCA architecture described in this paper similarly models decision making and learning as resulting from interactions among non-symbolic cognitive elements.

The idea of the mind as a complex, dynamic, sub-symbolic system is supported by neuroscientific evidence that populations of neurons, instead of single cells, encode mental representations (see [**62**] for examples). Dynamic systems theory can explain puzzling results in developmental psychology, such as counterintuitive performance by infants on the A-not-B task [**61**], in which children with experience in reaching toward one container for a hidden toy consistently reach toward that same container even after observing the toy being hidden in a different container. The development of a shape bias [**56**], word comprehension [**31**], deferred imitation, self-recognition [**24**] and a number of other cognitive changes in the first years of life have also been explained by dynamic systems models.

CHAPTER 2

Agent Structure

Hybrid dynamical cognitive agents (HDCAs) are modeled as *hybrid dynamical systems*, using *beliefs*, *desires*, and *intentions* connected in a *spreading activation network* to represent goal-directed cognitive systems. This chapter presents technical descriptions of these structures that underlie HDCA models.

2.1. Hybrid Dynamical Systems

A hybrid dynamical system is defined by a set of states, or *modes*, which are connected to other modes by *transitions*. Modes have both continuous and discrete parts: Continuous parts evolve according to systems of differential equations that define changes of system variables called *elements* over time; discrete parts of modes remain static during system evolution, but are changed instantaneously by transitions to other modes. Discrete parts of modes can be seen as "identifiers" that specify which mode the system is in; the system's continuous evolution can be different in each mode, and therefore the evolution of continuous parts depends on the discrete identifier.

Modes also contain one or more *guards* to monitor transitions; each mode-specific guard has a set of *conditions*, boolean statements about values of the system's elements, and a single transition that specifies a new mode to enter. When the continuous part of a mode matches conditions in one of its guards, that guard is said to be *tripped* and the associated transition is taken, instantaneously changing the discrete part of the state to identify which new mode has been entered. Continuous elements of the system may also be changed instantaneously and discretely by *side effects* during a transition.

The formal definition of the hybrid dynamical system used in this thesis (based on the definition in [**11**]) is provided here for mathematical interest, but is not essential for

understanding HDCA behavior. In this thesis, a hybrid dynamical system is a six-tuple $H = (V, n, X_0, F, Inv, R)$ where:

- V is a finite set of *locations*, the "identifiers" for each mode.
- $n \geq 0$ is a nonnegative integer called the *dimension* of H. The state space of H is $X = V \times \mathbb{R}^n$. Each state has the form (l, x) where $l \in V$ is the *discrete* part of the state, and $x \in \mathbb{R}^n$ is the *continuous* part.
- $X_0 \subseteq X$ is the set of initial states.
- $F : X \to \mathbb{R}^n$ assigns to each state $(l, x) \in X$ a value $F(l, x) \in \mathbb{R}^n$ that constrains the time derivative of the continuous part of the state. Thus in discrete location l, the continuous part of the state satisfies the *differential equation* $\dot{x} = F(l, x)$. [1]
- $Inv : V \to 2^{\mathbb{R}^n}$ assigns to each location $l \in V$ an *invariant* set $Inv(l) \subseteq \mathbb{R}^n$ that signifies appropriate values of the continuous part of the state while the discrete part is l.
- $R \subseteq X \times X$ is a relation capturing discontinuous state changes.

Hybrid systems can be modeled by *hybrid automata*, combinations of finite automata representing discrete, symbolic dynamics, and continuous-time state-space models describing continuous dynamics [**35, 66**]. As in [**11**], hybrid automata can be represented by finite directed graphs with vertices V and edges E where

$$E = \{(l, l') \in V \times V \mid ((l, x), (l', x')) \in R \; for \; some \; x \in Inv(l), \; x' \in Inv(l')\}. \quad (1)$$

A *guard* set on each edge $e \in E$ is defined as

$$Guard(e) = \{x \in Inv(l) \mid ((l, x), (l', x')) \in R \; for \; some \; x' \in Inv(l')\} \quad (2)$$

and a set of *side effects* (called a reset map in [**11**]) is defined as

$$SideEffect(e,x) = \{x' \in Inv(l') \mid ((l, x), (l', x')) \in R\}. \quad (3)$$

[1] This differs from the Alur et al. model [**11**] by constraining the continuous part to a differential equation instead of a differential inclusion $\dot{x} \in F(l, x)$.

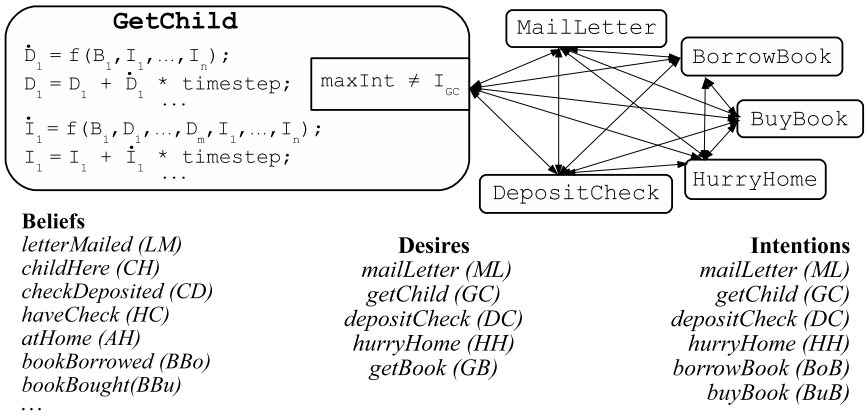

FIGURE 2.1. A hybrid dynamical system with one elaborated mode, and abbreviations for beliefs, desires and intentions. In the guard, maxInt is the current maximal intention.

A hybrid system begins in some initial state (l, x) in X_0. The continuous part of the state x evolves according to a differential equation $\dot{x} = F(l, x)$ while the discrete part l remains the same. When x meets any condition in a guard, that is, $x \in$ *Guard(e)* where $e = (l, l')$ for some $e \in E$, a jump is taken to a connected state (l', x') such that $((l, x), (l', x')) \in R$, with a new continuous state $x' \in$ *SideEffect(e,x)*.

A partially specified hybrid automaton is illustrated in Figure 2.1; modes are rectangles with rounded corners, guards are rectangles with 90° corners, and transitions are arrows from corresponding guards to modes. This diagram illustrates the hybrid dynamical system structure for the HDCA in simulations described in Chapter 5. In this figure, mode `GetChild` is expanded to show differential equations that describe the continuous part of that mode and the guard[2] for transitions out of that mode. The structure of these differential equations is described in more detail in section 3.1.2.

In summary, hybrid dynamical systems are models for systems in which continuous dynamics depend on the systems' current modes. For HDCAs, cognitive elements such as beliefs, desires and intentions comprise continuous parts of modes; discrete parts

[2]According to the technical definition, there is a different guard for each transition from `GetChild` to another mode; in the current implementation, a single guard is parameterized for each transition.

of modes are the names of each action agents can perform. Guards contain one or more conditions on cognitive elements; in Figure 2.1, for example, the guard condition is that the maximal intention—the intention with the highest activation—is not I_{GC}. Transitions from one mode to another represent switching from one action to another; on transitions, the discrete part (the name of the mode) is changed instantaneously. Additionally, each mode can have its own system of differential equations that define cognitive evolution; intuitively, agents' cognitive systems evolve differently depending on which action they're currently performing.

2.2. Beliefs, Desires, Intentions

The *Belief, Desire, Intention* (BDI) model ([**30**] and others) identifies three elements involved in decision making: *beliefs* are knowledge of the world; *desires* are goals, either states of the environment or sequences of actions that an agent wishes to achieve; *intentions* are commitments to goals. This theory, described by [**17**], has been widely applied to intelligent agent decision making (see section 1.2 for examples).

In BDI models, agent behavior results from interactions among the three elements, which are typically represented propositionally. In the HDCA models described by this thesis, however, BDI elements are ascribed continuous *activation values* ranging from -10 to $+10$ that indicate elements' *salience*; larger absolute values represent higher salience, that is, elements being more "in mind" or felt more strongly, and negative values represent negations of elements. For instance, a value of $+8$ on desire D_{GB} means that an agent is highly aware of its desire to attain a book, whereas a value of -3 means that the agent only slightly feels the desire not to get a book (see Figure 2.1 for BDI abbreviations). BDI elements are implemented in the hybrid dynamical system framework for HDCAs as elements of continuous parts of modes; BDI elements are state variables whose continuously evolving activations are determined by mode-specific differential equations. Dynamics of BDI activations are described further in section 2.3.

To make agents capable of perception for learning, an additional structure is implemented alongside beliefs, desires, and intentions. *Ground concepts* represent aspects of agents' environments [**1**], such as target locations for actions, and are used for perception: Perceptual inputs activate appropriate ground concepts which can then be used to learn information about the environment (as in Chapter 4). These additional elements are only applied during agents' training phases, and therefore are only discussed in learning contexts.

In the hybrid dynamical system that models HDCAs in this thesis, continuous parts of modes consist of continuously evolving beliefs, desires, and intentions (and ground concepts for learning); discrete parts are potential actions, as indicated in Figure 2.1. Modes have systems of equations governing continuous evolutions of cognitive BDI elements over time; these systems of differential equations may differ between modes. For instance, activation on one belief might increase more rapidly as time passes when an agent is in one mode over another; two related intentions might have more strongly inhibitory effects on each other in one mode than another. This evolution of cognitive elements is modeled by a spreading activation network.

2.3. Spreading Activation

Cognitive BDI elements are nodes in *spreading activation networks*; each element has links to related elements over which activation can spread. Elements' activation values, which represent salience, are therefore determined partly by the amount of activation spread from related elements. Spreading activation networks can be seen as abstractions of neuronal connections: In the brain, neurons are activated when their electrochemical membrane potential increases above a certain threshold. These activated neurons can "fire," eliciting raised membrane potential in connected neurons. Similarly, in spreading activation networks in this thesis, activations encode cognitive function and determine behavior. The network described here is different from a neural network, however,

because each node represents a coherent belief, desire, or intention, rather than some lower-level unit as neurons do in the brain.

Spreading activation networks in this thesis are encoded by systems of differential equations in each mode (see section 2.1). Activation spreads from *originator* nodes to *receiver* nodes by altering the rate of change of activation levels for receiver elements. For example, suppose f is an element in the spreading activation network, and $G = \{g_1, \ldots, g_n\}$ is the set of elements with spreading activation links to f with corresponding weights w_i. Then

$$\dot{f} = \sum_{i=1}^{n} w_i \cdot g_i \tag{4}$$

is the rate of change of activation on f, and

$$f_t = f_{t-1} + \Delta f = f_{t-1} + (\dot{f} \cdot s) \tag{5}$$

is the value of f at time t, where s is the length of a timestep in the simulation, f_{t-1} is the activation value of f at the previous timestep, and Δf is the change in activation level of f due to spreading activation from related elements. (Equations 8 and 9 provide BDI-specific examples.) Negative activation can also spread along associative links; in this case, activations of receiver elements decrease proportionally to the activation of originator elements. For instance, negative activation on belief B_{HL} decreases activation on intention I_{ML}, because the agent will not commit to actions it does not believe it can achieve (see Figure 2.1 for BDI abbreviations). Negative activations spreading along negatively weighted connections increase activations in receiving elements: belief B_{LM} has a negatively-weighted link to D_{ML}, so negative activation on B_{LM} corresponds to a positive change in the activation level of desire D_{ML}.

In this implementation, beliefs are restricted to activation values of $+10$ or -10, and are not affected by spreading activation from other elements, although they can spread activation to other elements. Values on beliefs can still be changed discretely as side effects on transitions (see section 2.1). This simplification allows beliefs to act as memory, encoding truth or falsity (as far as the agent knows) of certain preconditions for action.

CHAPTER 3

Hybrid Dynamical Cognitive Agents

Hybrid dynamical cognitive agents (*HDCAs*) are intelligent, autonomous, embodied agents that perform goal-directed behavior in navigation-based domains. As described in Chapter 2, these hybrid reactive/deliberative agents use a BDI model to represent beliefs, desires, and *dynamical intentions*; BDI elements have activation values indicating salience, and are connected in a spreading activation network. HDCAs are modeled as hybrid dynamical systems: Continuously evolving BDI elements comprise continuous parts of modes; action names comprise discrete parts.

In the HDCA implementation for this thesis, HDCAs select actions to perform, such as `MailLetter` or `DepositCheck`, and navigate to complete these actions at different locations in a grid city, as pictured in Figure 5.1. HDCA behavior is determined by the state of dynamical intentions: Actions are in one-to-one correspondence with intentions, and agents perform the action indicated by the current maximally activated intention. Actions are sequenced into *plans* by arranging intentions in decreasing order of activation. *Dynamical re-planning* occurs when the order of intentions changes due to natural activation evolution (e.g., within the cognitive spreading activation network). *Deliberative re-planning* occurs when HDCAs' deliberative system discretely re-assigns intention activations (e.g., according to a utility score based on distance to targets and intention activations). The structure of HDCAs is illustrated in Figure 3.1. The functions of both reactive and deliberative systems, including the roles of dynamical intentions in each system, are described in this chapter.

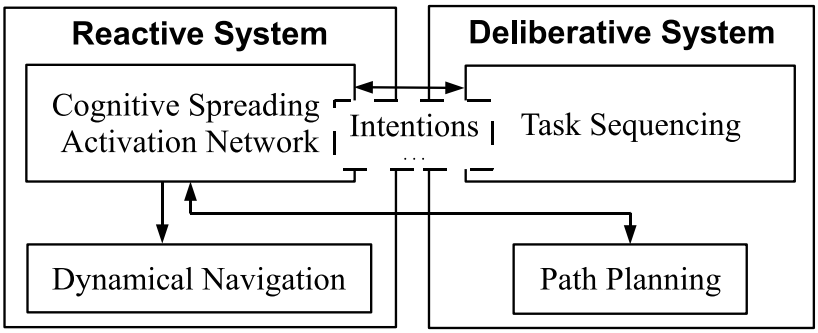

FIGURE 3.1. An architecture diagram detailing the roles of reactive and deliberative systems in an HDCA. Note that representations of dynamical intentions are shared by both levels.

3.1. Reactive System

Because HDCAs are hybrid reactive/deliberative agents, discussion about agent structure can be divided into separate discussions about the reactive and deliberative systems. This section describes HDCAs' reactive system, which includes the spreading activation network of BDI elements and a mechanism for dynamically navigating to targets. HDCA behavior is mostly controlled by continuous evolution of cognitive and physical elements in this reactive system.

3.1.1. Dynamical Navigation. HDCAs perform continuous, intersection-to-intersection navigation in a grid city by maintaining constant forward velocity and dynamically adjusting angular velocity based on the presence of targets and obstacles, as in [**5, 6**]. *Angular attractor* and *repeller* functions represent targets and obstacles, respectively, and the weighted contributions of these two functions is summed to determine the change in heading angle, $\dot{\phi}$, which dictates the direction of the agent's angular velocity. As defined in [**5, 6**], the angular velocity function is a non-linear dynamical system,

$$\dot{\phi} = f(\phi, \mathbf{env}) = |w_{tar}|f_{tar} + |w_{obs}|f_{obs} + n \tag{6}$$

where f_{tar} and f_{obs} are attractor and repeller functions, w_{tar} and w_{obs} are their respective weights in the system, and n is a noise term which helps prevent the system being

trapped at critical points. In the implementation described by this thesis, f_{tar} is set to be the next appropriate intersection in the grid city; intersection selection is discussed in section 3.2.2.

The weights in the heading angle function in equation 6 are determined by computing fixed points of the non-linear system

$$\begin{cases} \dot{w}_{tar} = \alpha_1 w_{tar}(1 - w_{tar}^2) - \gamma_{12} w_{tar} w_{obs}^2 + n \\ \dot{w}_{obs} = \alpha_2 w_{obs}(1 - w_{obs}^2) - \gamma_{21} w_{obs} w_{tar}^2 + n \end{cases}, \tag{7}$$

in which α and γ are parameters designed to reflect stability conditions for the system. Other parameters are hidden in this function, such as the amount of influence obstacles or targets have on agents, as described in [**32**]; they will not be further discussed here.

Previously [**5**], these hybrid dynamical navigating agents also performed rudimentary high-level decision making in the form of street selection. This method for dynamical street selection is broadened in HDCAs to produce more complex task selection by applying features of high-level human decision making, specifically intentions, to dynamical reasoning.

3.1.2. Cognitive Spreading Activation Network. Beliefs, desires, and intentions are thought to form the basis of human reasoning [**17**], and have been effective and popular ways of implementing decision making in artificial intelligence systems (see section 1.2 for examples). BDI models are traditionally implemented deliberatively, with logical proof-based reasoning about symbolically represented beliefs, desires and intentions.

In HDCAs, however, elements of BDI are represented with a dynamical system, allowing for reactive decision making. In particular, desires and intentions have activation values and spread activation to other desires and intentions (as described in section 2.3). Spreading activation networks are implemented as differential equations, such as these two examples for the desire to deposit a check D_{DC} and intention to deposit a check

I_{DC}, which have several terms elided for brevity:

$$\dot{D}_{DC} = a_1 B_{HC} + a_3 I_{DC} - a_5 I_{GC} + \ldots \quad (8)$$

$$\dot{I}_{DC} = b_1 B_{HC} + b_3 D_{DC} - b_6 D_{HH} + b_8 I_{DC} - b_{10} I_{GC} + \ldots \quad (9)$$

The terms in these equations represent influence from other BDI elements, and coefficients a_i and b_i encode strength of spreading activation links. Each coefficient may encode more than a single numerical value; specifically, coefficients of intention terms contain additional functions that encode properties of intention described in section 3.3. Coefficients are initially pre-specified for each simulation run, although Chapter 4 details how they can be changed dynamically through learning in order to encode typically deliberative information, such as distances between targets.

Recall from section 2.1 that HDCAs are implemented as hybrid dynamical systems, with each mode encoding a different action. *Action selection* for HDCAs occurs when agents' cognitive systems transition into new modes. These transitions are triggered when certain elements of the continuous system—the BDI elements—meet conditions in mode-specific guards. In this implementation, guards are tripped when the maximal intention (i.e., the intention with the highest activation value) is no longer the intention for the action of that mode. For instance, in Figure 2.1, a new action is selected (a transition is taken) in mode `GetChild` when the maximal sequencing intention is no longer I_{GC}. In this way, HDCA behavior is instantaneously responsive to changes in system state; as in traditional reactive systems, HDCA action selection does not require explicit symbolic reasoning.

Dynamical intentions are responsible for mode transitions, but other BDI elements affect dynamical intentions through spreading activation links, and therefore play some role in action selection as well. Spreading activation connections are predefined to support desired outcomes; in equation 9, for instance, belief that an agent has a check, B_{HC}, and desire to deposit a check, D_{DC}, both spread positive activation to intention to deposit check, I_{DC}. On the other hand, conflicting intention to get the child, I_{GC}, has an inhibitory link with I_{DC}. Because HDCAs are built from a dynamical framework,

with continuous functions modeling agent state, reasoning can be arbitrarily precise: Change in activation levels can be measured at any desired resolution, although this feature is not fully exploited in the implementation for this thesis.

In traditional intelligent agents, *plans* are sequences of actions that achieve goals [**55**]. By this definition, plans for HDCAs are sequences of modes; however, HDCAs do not explicitly select which modes to enter, but rather transition among modes in response to continuously evolving system state. Therefore, a different definition of "plan" is required for HDCAs: Plans in this framework are series of intention activations that lead agents to perform intended sequences of actions under normal environmental conditions[1] [**1**]. These activations are expected to evolve to meet guard conditions and cause transitions to intended modes (i.e., cause agents to dynamically select particular actions) in the intended order. In this simplified model, a plan is a sequence of intentions ordered by descending activation value, which is expected to lead to ordered performance of actions.

Reactive re-planning occurs when existing plans are no longer beneficial or other options become better, often due to unexpected changes in environments. Reactive re-planning is particularly important in stochastic environments where environmental conditions might change before agents can finish a logical reasoning process; by implementing reasoning dynamically, HDCAs' cognitive systems instantaneously respond to changes in knowledge, goals, and commitments, and thus can avoid reasoning with outdated information in rapidly-changing environments. Additionally, as suggested by cognitive dynamical systems theories [**61**], small cognitive changes can eventually lead to large behavioral effects; these small changes can be represented to any degree desired in this dynamical HDCA model.

Traditionally, purely reactive agents such as [**18**] have no memory—they do not store information gained in the past—although efforts have been made to include some forms of memory, such as representations of an environment's layout [**41**]. A form of

[1]Although some traditional plans include contingencies for unexpected events, HDCAs' plans do not; agents must re-plan if environmental conditions alter expected activation evolution.

memory exists for HDCAs, however, encoded by activation levels on beliefs, which are propositional statements. Specifically, beliefs are used as binary "yes/no" variables: An activation level of $+10$ indicates affirmation of the belief and a level of -10 indicates negation of the belief. In the HDCA implementation described by this thesis, belief activations are held constant and only change discretely through side effects during transitions.

3.2. Deliberative System

HDCAs' deliberative systems augment reactive behavior with explicit reasoning and plan generation, using information that is not readily encoded reactively, such as knowledge about distances between targets or about domain-specific rules. The deliberative system is invoked during mode transitions in response to certain conditions: when agents are unexpectedly interrupted while completing actions; or when there is uncertainty about which action to perform next. Deliberation is also used to find the best path to a target using distance information.

3.2.1. Task Sequencing. Deliberative systems complement reactive system behavior, such as re-planning and action selection, by explicitly reasoning about actions in conditions that cannot be handled well reactively. Specifically, two types of conditions will invoke deliberation for task sequencing. The first is an unexpected external event—in this implementation, a blockaded street—that prevents the agent from carrying out its actions as intended. In such cases, agents deliberate using the newly acquired information that prompted deliberation (the position of the blockade) and may alter their task sequences as a result. The second condition that invokes deliberation is internal uncertainty; in this implementation, if the two highest intention activations are within a threshold value of each other when it is time to select an action (such as when a previous action has just been completed), the agent is uncertain about which next action to select, and it will deliberate using additional information that is not encoded reactively.

The result of deliberation is a plan, an ordered sequence of actions. To generate plans, the deliberative system first evaluates which actions are incomplete, based on beliefs about actions having been completed; only actions a with -10 activation on belief that a has been completed are considered further. These incomplete actions are scored based on current intention activations and distance to the target for that action. A score for action a is calculated as:

$$S_a = I_a \cdot c - dist(t_{a-1}, t_a) \tag{10}$$

where I_a is activation on the intention to perform action a, $dist(t_{a-1}, t_a)$ is the distance from the target t_{a-1} of the previous action in the plan (or from the agent's current position, if a is the first action in the plan) to the target of a, and c is a predefined scaling factor.

These incomplete actions are arranged in decreasing scored order to form a plan. To execute this plan on the reactive level, intention activations for all actions in the plan are instantaneously set so that the first action has the highest intention activation, the second has the second highest, and so on, according to the following formula:

$$I_a = \frac{m}{i} \tag{11}$$

where i is the position of action a in the plan, and m is the predefined maximum intention value. Thus, intention to do the first action in the plan is assigned maximum activation value m, intention to do the second action is assigned activation value $m/2$, and so on. Based on this formula, the difference between activation values of successive intentions decreases for intentions later in the plan. This assignment of activations with successively decreasing differences is intuitively reasonable: actions planned to be completed further in the future have more time to be resequenced—and therefore their ordering is less certain—than actions planned to be completed sooner.

Once new activation values are assigned to planned intentions, the reactive level resumes execution and completes the plan dynamically, as described in section 3.1. The only exception to this process occurs if an agent is within half a street-length of (i.e., is

on the same block as) a target for an incomplete action; in that case, the agent enters the mode for that action immediately and completes that task first, without generating a plan.

3.2.2. Path Planning. In the grid-city domain described by this thesis (and illustrated in Figure 5.1), agents set intersections as *sub-targets*, intermediate navigational steps on the way to targets for actions. Deliberation is invoked at each intersection to select the next best direction for travel; between sub-targets, agents navigate dynamically as in section 3.1.1. Calculating the next best intersection is a deliberative task because it manipulates knowledge—locations of targets—that is not immediately perceptible and therefore not available to the reactive system.

A moving blockade poses additional navigational challenges for HDCAs. This blockade can restrict access to one street at any time; its position changes arbitrarily, presenting a dynamic element of the environment. If a blockade is perceived to be blocking the street between an agent's current position and the next best intersection, the agent invokes deliberation about task sequences as in section 3.2.1. This deliberation accounts for the blockade when calculating the distance between t_{a-1} and t_a for its current intended action. Updated knowledge about blockaded streets might lead to an altered plan by decreasing the score for the action whose target is blocked by the blockade.

3.3. Properties of Intention

Intentions in BDI models are based on Bratman's philosophical definition [**17**]. In particular, he describes intentions as elements of decision making that are equally important to beliefs and desires, but that are distinct from desires in a number of ways: once intentions are formed (i.e., once a commitment to action is formed), they tend to persist; the presence of one intention tends to eliminate the adoption of a conflicting intention; and intentions, not desires, are responsible for determining behavior.

In traditional BDI applications (see section 1.2.2 for examples), intentions are adopted after a logical reasoning loop; adopted intentions represent actions to which agents have

commitments. Dynamical intentions for HDCAs, however, are modeled with activation levels that indicate intentions' salience to agents, and can thus be seen as always adopted to some greater or lesser degree. It is necessary, therefore, to modify Bratman's definition slightly to suit this dynamical framework: Instead of *adopted* intentions persisting, resisting conflict, and controlling conduct, it is now the case that *high active* intentions resist reconsideration, avoid conflict, and control conduct. It is currently unclear how (though not necessarily impossible) to reactively extend other, more future-oriented features of intentions; for instance, agents track the success of their intentions and tend to repeatedly attempt completion in the case of failure, but this type of future-directed reasoning is not straightforwardly implemented in the current HDCA model.

The three reactive properties of dynamical intentions are elaborated upon in the following three sections.

3.3.1. Reconsideration resistance. Intentions are commitments to action [**17**]; intuitively, once commitments are formed, they tend not to disappear. Dynamical intentions persist by means of a *persistence factor* PF, which is part of coefficients for intention terms in differential equations for intentions (b_i in equation 9). That is, for every intention I_a, the differential equation $\dot{I}_a$ includes the following term:

$$\dot{I}_a = \ldots - k_n \cdot PF(I_a) \cdot I_b \ \ldots \tag{12}$$

where I_b is another intention and $a \neq b$. In equation 9, for example, the coefficient of term I_{GC} in the equation for $\dot{I}_{DC}$ is $b_{10} = k_{10} \cdot PF(I_{DC})$.

Persistence factor is designed so that higher activated I_a minimizes inhibitory effects of conflicting intention I_b on the activation level of I_a:

$$PF(I_a) = 1 - \frac{|I_a|}{\sum_i |I_i| + \epsilon}, \tag{13}$$

where i ranges over all intentions and noise term $\epsilon > 0$ avoids division by zero. As activation of I_a increases relative to other intentions (that is, as the value of the fraction approaches 1), coefficients of terms for other intentions are multiplied by proportionately decreasing values. Since other intention terms subtract activation from $\dot{I}_a$, multiplying

them by decreasing values lessens their inhibitory effect on $\dot{I}_a$. Note also that the effect of $PF(I_a)$ diminishes when other intentions are highly active relative to I_a.

3.3.2. Conflict avoidance. Because intentions are commitments to perform actions, and not just desires for actions, a rational agent's adopted intentions do not conflict [**17**]. In HDCAs, however, dynamical intentions are always activated to some degree; in this system, *high-active* intentions tend not to conflict, with more active intentions avoiding conflict more effectively than less active intentions.

As with reconsideration resistance, conflict avoidance is implemented as part of coefficients for intention terms in differential equations for intentions. *Non-conflict factor* NCF is an element of the coefficient for each intention term in the differential equation for I_a; for example, equation 12 can be decomposed to

$$\dot{I}_a = \ldots - m_n \cdot NCF(I_b) \cdot PF(I_a) \cdot I_b \ \ldots. \tag{14}$$

For NCF, there is no restriction that $a \neq b$, so the term for I_a in the equation for $\dot{I}_a$ is also multiplied by NCF; that is, the differential equation $\dot{I}_a$ includes a term $\ldots - m_p \cdot NCF(I_a) \cdot I_a \ \ldots$.

Non-conflict factor increases inhibitory influence of highly active intentions on other intentions in the system. Specifically, NCF is implemented as

$$NCF(I_b) = 1 + \frac{1.6 \cdot I_b^8}{10^8} + \frac{0.8 \cdot I_b^9}{10^9}. \tag{15}$$

NCF was hand-tuned to be nearly 1 for all $|I_b| < 5$, with steep increases at extremely high activation values (reaching 3.4 as I_b nears 10) and extremely low activation values (reaching 1.8 as I_b nears -10). Therefore, $NCF(I_b)$ is only effective when the activation of I_b is very high or very low. Values of the constants can be changed to alter this threshold.

3.3.3. Conduct control. In Bratman's definition [**17**], intentions are distinct from desires because intentions, not desires, control behavior. To paraphrase Bratman's example, suppose an agent has the desire for a milkshake and the desire to lose weight. Upon entering the cafeteria, this agent notices that milkshakes are being served; it has

already committed to a weight loss program that excludes sugary drinks, however, and therefore the agent's intention to lose weight (its commitment to the program) precludes fulfilling its desire to drink a milkshake.

In the current HDCA model, agents' behaviors are dictated by their most highly activated (i.e., maximal) intention. Agents will perform an action until the intention to perform that action is no longer the maximal intention, either because the goal of that action was completed or because some other intention has naturally evolved a higher activation. Guards within modes check agents' intention activations at every timestep, and activate transitions to appropriate modes if maximal intentions are no longer the intentions to perform current actions.

CHAPTER 4

Learning

Learning is a process of acquiring and consolidating information for later retrieval. Learning for autonomous agents is important for building robust, adaptive agents: Autonomous agents that learn new information or new behaviors can adapt to novel and unexpected situations; autonomous agents with pre-specified behaviors can perform only as well as their designers foresee.

Neuroscientific theories of learning have long posited that synaptic changes between neurons or systems of neurons are responsible for permanence of memories. In 1893, Ramon y Cajal (as noted in [**16**]) proposed that learning builds new synaptic connections. Hebb [**33**] expanded on this by theorizing that *reverbatory traces*, cycles of activations along hierarchically interconnected structures of neurons, temporarily encode memories until synaptic changes can permanently encode them. This form of learning through modification of neuronal connections can be applied to the computational spreading activation network for HDCAs.

Computational models like *neural networks* [**42, 54**] also apply neuronal ideas to computer systems: In these models, knowledge is encoded by distributed activation across nodes of a network; learning in connectionist systems involves adjusting weights between nodes to change distributed knowledge representations. The framework described by this thesis is inspired by such connectionist architectures, although in HDCAs' simplified model for knowledge representation, knowledge is encapsulated in individual nodes rather than being distributed across multiple nodes. Learning for HDCAs is similar to learning for connectionist architectures because weights of connections are adjusted in response to experience; HDCA learning is different from connectionist learning, however, because a single connection between two nodes, rather than a distributed

system of nodes, encodes a relation between two items of knowledge. This chapter details one type of HDCA learning and briefly describes two other potential learning methods.

4.1. Hebbian Learning

Donald Hebb [**33**] postulated that a (pre-synaptic) neuron repeatedly firing to another (post-synaptic) neuron might cause structural changes leading to increased efficiency in activating the post-synaptic neuron by firing the pre-synaptic neuron. This idea has been generalized to a type of learning called *Hebbian learning*, in which elements of a spreading activation network develop stronger connections to other elements with repeated co-occuring activation.

In the current implementation, HDCAs learn to reactively account for relative distances between targets by altering connections between intentions, which increases the likelihood that agents will intend to perform one task when a nearby task is strongly intended. Hebbian learning adjusts coefficients of intention differential equations (for example, equation 9) by decreasing coefficients for intention terms (which are subtracted in the equations). During a training session with Hebbian learning, negative spreading activation connections are weakened between intentions with proximate tasks[1], while connections remain unchanged between intentions for tasks whose targets are distant. As a result, a highly intended task spreads less negative activation—has less of an inhibitory effect—on intentions for tasks with nearby targets.

During training sessions, agents represent targets as *ground concepts*, elements in the cognitive spreading activation network (see section 2.2). Ground concepts begin with baseline activation values of 0, and become instantaneously activated when agents navigate to within a *radius of perception* of targets. This instantaneous activation can be formally defined as a hybrid dynamical system transition from a mode to itself, with a side-effect of increased activation on the ground concept in question (section 2.1

[1]Hebb's ideas [**33**] focused on strengthening activation connections, but Hebbian learning can also be generalized to weakening activation connections in response to co-activation.

describes the underlying hybrid dynamical system). When agents move out of a target's radius of perception, activation on the target's ground concept decreases gradually to baseline level; this is modeled in the hybrid dynamical system using differential equations for continuous evolution.

Hebbian learning associates intentions for actions whose targets have concurrently co-active ground concepts. At every timestep during a training session, for ground concepts a and b with activations $\alpha, \beta > 0$ respectively, the following learning algorithm is applied:

$$IC(I_a, I_b) = IC(I_a, I_b) - \frac{\beta}{c_1}. \tag{16}$$

In this equation, $IC(I_a, I_b)$ is the coefficient on the term for intention I_b in the differential equation for intention I_a (that is, in $\dot{I_a}$), and c_1 is a scaling constant. Targets, their ground concepts, and intentions are all in one-to-one correspondence, so ground concepts a and b map directly to intentions I_a and I_b. For example, in $\dot{I_{DC}} = \ldots - b_{10} I_{GC} + \ldots$, which is part of equation 9, b_{10} is decreased by β/c_1 if activation on the ground concept for the school, where I_{GC} can be completed, is above baseline at the same time as activation for the bank, where I_{DC} can be fulfilled.

Because all intentions in this system conflict, intentions have negative spreading activation links among them; this Hebbian learning algorithm is applied to intentions each time two ground concepts have above baseline activation. As a result of learning, intentions for proximate actions tend to become highly activated together, and agents consecutively perform actions whose targets are near each other. See section 5.4 for an illustrative simulation.

4.2. Other Types of Learning

The HDCA framework has potential for types of learning that have not yet been fully explored. Two such learning methods—*conceptual* learning and *behavioral* learning—are described in this section.

4.2.1. Conceptual Learning. *Conceptual learning* is a form of classical conditioning; associations between two stimuli are learned through iterated exposure to both stimuli concurrently. A simple example of conceptual learning has been implemented in an environment with stimuli such as `park` and `friend` [**7**]. Over repeated visits to the park to see its friend, an agent learned an association between `park` and `friend`; eventually, simply arriving at the park "reminded" the agent of its friend regardless of whether `friend` was present.

Conceptual learning functions in a manner similar to Hebbian learning: spreading activation links are adjusted over repeated exposure to co-active elements. In conceptual learning, however, links between ground concepts are strengthened, instead of links between intentions being weakened. In training for conceptual learning, as in training for Hebbian learning, ground concepts become activated when agents enter a radius of perception from stimuli. At every timestep, for every pair of ground concepts activated above a pre-specified threshold, if spreading activation links exist between these concepts then these links are strengthened proportionally to the concepts' activations. Similarly, strengths of links between concepts whose activations are below a certain threshold are weakened proportionally. Weakening links for ground concepts that are not co-active ensures that concepts co-activated by chance or for a short time do not become permanently associated, and that concepts that were once co-active but no longer occur together eventually have their associations removed.

Conceptual learning creates stimuli associations so that one stimulus becomes salient when a second stimulus is salient, even in the absence of the first stimulus. In [**7**], as described above, a conceptual learning agent builds an association between `friend` and `park`: Before learning, a weak spreading activation link causes `friend` activation to increase slightly whenever `park` is activated; after learning, `friend` becomes highly salient whenever `park` is activated, even if `friend` is not directly perceived, due to spreading of activation over the strengthened link.

4.2.2. Behavioral Learning. A third kind of (yet unimplemented) HDCA learning builds associations between behaviors, rather than between intentions or ground concepts. *Behavioral learning* alters guards, sets of conditions which mediate transitions between modes (see section 2.1). By altering conditions that regulate when transitions are taken, behavioral learning can change how one behavior leads to another.

For instance, suppose that on its drive to work, an agent regularly makes a left turn at one intersection followed by a right turn at the next. After learning through repeated experience, this sequence of turns becomes associated so that the agent no longer actively considers which turns to take to get closer to its target. This behavior association becomes apparent, for instance, when the agent intends to go a different direction but instead performs the turn sequence as on its way to work, because the learned association between behaviors is so strong.

Behavioral learning changes conditions within guards in response to repeated tripping of the guard. Every time conditions on a guard are met and that guard is tripped, the conditions can be adjusted to be slightly more inclusive; for instance, if the guard requires that the value of some cognitive element be less than a threshold d, the value of d could be increased by 10% every time that condition is met, in order to include more possible values of the cognitive element. This change makes it more likely that the system will transition from one mode to another, associating the behaviors in those two modes. An implementation of behavioral learning would open the possibility for new and interesting behavioral dynamics, such as skill learning, in which actions that are practiced become easier to perform.

CHAPTER 5

Demonstrations and Experiments

Simulations of HDCA behavior described in this chapter demonstrate roles of the reactive and deliberative systems, and highlight how behavior is affected by a shared representation of dynamical intention. In the following simulations, unless otherwise noted, agents navigate the 4-block by 4-block grid city environment illustrated on the left of Figure 5.1. Navigation is based on intersection selection described in section 3.2.2 and dynamical navigation described in section 3.1.1. Agents complete tasks at various locations in the grid city by navigating to associated task targets, identified by two-letter abbreviations in Figure 5.1; when agents reach each target location, tasks are completed instantaneously. The order in which agents complete tasks depends on activations of agents' intentions at every timestep; agents' desires and intentions are represented at the right of Figure 5.1 as colored circles, where larger diameter means greater positive activation.

These demonstrations begin with a single agent on the leftmost side of the grid city. Agents have initial desire and intention activations as specified in Table 5.1; unless noted, agents initially believe that all tasks are incomplete except for `MailLetter`. Demonstrations discussed in sections 5.1, 5.2 and 5.3 have been published [**2**] and movies and data relating to the demonstrations are available at the supplementary website for that paper [**4**].

5.1. Deliberative Task Sequencing

One simulation of an agent completing tasks shows both deliberative task sequencing (as in section 3.2.1) and reactive re-planning (as in section 3.1.2). The first of these behaviors is described in this section; reactive re-planning is described in section 5.2.

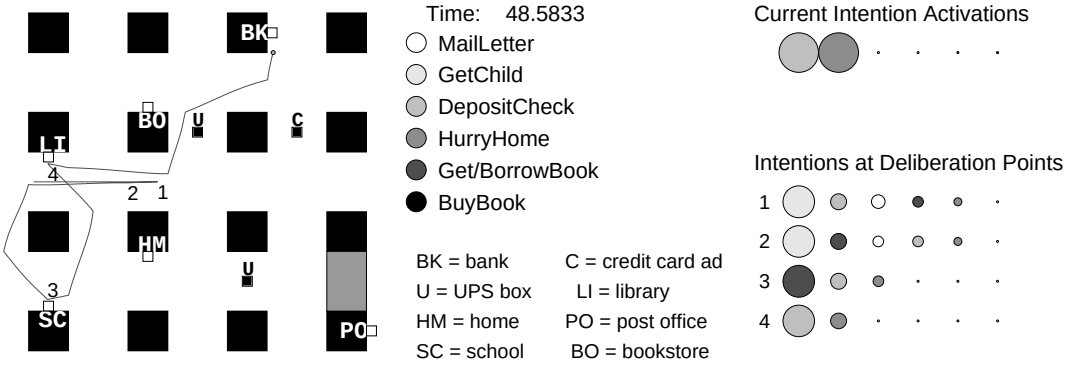

FIGURE 5.1. Four-block by four-block grid city environment with a trace of a single agent's path. Targets of actions are white squares abutting black buildings, and a moving blockade is the gray box blocking one street. A key provides abbreviations for targets. The agent's path trace indicates that it has gone to the school and library and is almost at the bank.

The agent in this simulation deliberates in response to two types of situations: an unexpected event interrupts the agent's current task (e.g., its path is blocked by a blockade); or the agent must select a new task from among intentions with similar activation values. In this simulation, intentions are similar when the difference in their activation values is less than two (though see section 6.1 for potential modifications).

During deliberation, the agent finds the "utility" of action a by subtracting the Manhattan distance to the target of a from the activation of the intention (multiplied by some scaling factor) to do a (as in section 3.2 and equation 10). The current hand-tuned scaling factor (variable c in equation 10) is 40, although this could be adjusted to alter how much the score is influenced by intention activation versus distance to target.

The agent begins in this simulation with similar intentions on actions `GetChild` and `DepositCheck`, although I_{DC} is slightly higher (as indicated in Table 5.1). During the simulation run, the agent deliberates at four points labeled in Figure 5.1; the right side

	ML	GC	DC	HH	GB/BoB	BuB
Desires	3	9	8	2	1	—
Intentions	6	9.3	10	2	5	-1

TABLE 5.1. Initial intention and desire activations for re-planning demonstrations

of the figure shows intention values after deliberation at each of these points. Initially, the agent begins by navigating toward the bank, but because the desire to `GetChild` is greater than the desire to `DepositCheck`, I_{GC} increases in activation more rapidly than I_{DC}. At point 1, activation on I_{GC} has exceeded activation on I_{DC}; this meets the criteria for uncertainty that prompts deliberation, because the two activations are within deliberation threshold (as explained in section 3.2.1). The agent deliberates at this point and, based on calculations described in equation 10, decides to complete `GetChild` first. It instantaneously reassigns intention activations based on scores determined during deliberation, as in equation 11; these new values are displayed on the right in Figure 5.1.

After this first deliberation, the agent continues navigating toward the school, but finds its path blocked by the moving blockade at point 2 in Figure 5.1. (In the figure, the blockade has again moved and is now in the lower right corner of the world.) This interruption causes the agent to deliberate again; it reorders I_{ML} and I_{BoB} but leaves the first task on the list unchanged, thus not perceptibly changing its immediate behavior. The agent finds a path around the blockade, eventually gets to the school and completes `GetChild`. At this point (point 3 in Figure 5.1), the next two highest intentions, I_{BoB} and I_{DC}, have a difference in activation value of less than 2 (the deliberation threshold), causing the agent to deliberate a third time. The agent successfully completes `BorrowBook`, and once again deliberates because the two highest intentions, I_{DC} and I_{HH}, are within the deliberation threshold value of each other.

This complex re-planning behavior arises from interactions among continuously evolving beliefs, desires, and intentions. Deliberation is only called when necessary,

and cleanly integrates with reactive behavior by instantaneously re-setting intention activations that the reactive system can act upon.

5.2. Reactive Re-Planning

Continuously evolving intention activations determine agents' behaviors: The most activated intention dictates which action the agent is currently performing, and reactive re-planning reorders intentions when their activation values change relative to other intentions (as described in section 3.1.2). As shown in this demonstration, this type of reactive action re-sequencing can correct erroneous plans resulting from inconsistent intentions and beliefs, without requiring deliberative reasoning. The simulation described in this section is the same simulation used for section 5.1, available at [**4**].

In this simulation, the agent begins with the belief that it does not have a letter (i.e., $B_{HL} = -10$) and an (inconsistently) positive activation on its intention to mail the letter, I_{ML}. The reactive system adjusts for this inconsistency through natural evolution of activation in the cognitive spreading activation network. Because $B_{HL} = -10$, and the rate of change of I_{ML} depends partly on the activation of B_{HL} (that is, $\dot{I}_{ML} = \ldots + k_i B_{HL} \ldots$), activation on I_{ML} decreases steadily over time, in response to the negative influence from B_{HL}. Furthermore, I_{ML} is never included in deliberation because of the belief that the task has been completed, and therefore never gets instantaneously adjusted by deliberation. Thus, the conflict between belief and intention is handled by a graceful decline of activation, rather than by explicit reasoning as would happen in the deliberative system. The gradual decrease of I_{ML} is apparent in the decreasing size of the circle representing that intention, in the time series on the right in Figure 5.1.

5.3. Properties of Intention

The reasoning system described in this paper is modeled on Bratman's theory of intention [**17**], which establishes *intentions* as distinct from *desires*, with several specific distinguishing properties: intentions *control conduct*, they *resist reconsideration*, and they *avoid conflict* with other intentions (as described in section 3.3). In this section,

	ML	GC	DC	HH	GB/BoB	BuB
Desires	1	10	10	3	3	—
Intentions	3.1	2	1	1	1	1

TABLE 5.2. Initial desire and baseline intention activations for properties of intention experiments

dynamical intentions are shown to have these properties, and therefore shown to be consistent with Bratman's definition of intention, which is the traditional definition for other BDI models (see section 1.2.2 for examples).

The experiments described in this section test for conflict avoidance and reconsideration resistance by arranging different initial desire and intention activations (specified in Table 5.2) so that one intention, I_{ML}, shows effects of both properties. Initial intention activations are set according to the baseline level indicated in Table 5.2, but vary in magnitude for different agents within the same simulation, displaying a range of conflict avoidance and reconsideration resistance effects. In these experiments, movement capabilities are removed, so agents do not navigate or complete tasks; instead, their cognitive systems evolve without being instantaneously re-set in response to task completion. Activations of I_{ML} were regularly sampled and recorded for each agent during one run of each simulation.

In each experiment, agents had identical initial desires, indicated in Table 5.2. Initial intention activation values for each agent A_i were $i/3$ times the baseline values indicated in the table (for example, for agent A_2, I_{GC} initially had an activation value of $2 \times (2/3) = 4/3 = 1.3333$).

5.3.1. Intentions avoid conflict. One feature of intentions, according to Bratman [**17**], is that intentions restrict the adoption of conflicting intentions. In our system, intentions conflict if greater (positive) activation on one intention diminishes activation

on the other; currently, every intention in the system conflicts with every other intention, and this relationship is encoded with negative terms in intentions' differential equations. As described in section 3.3.2, non-conflict factor NCF encodes intentions' conflict resistance by causing highly-activated intentions to have greater negative effects on other intentions, inhibiting the salience of competing intentions.

The non-conflict factor is applied to intention terms in differential equations for intentions: For every intention I_a, every term encoding intention I_b in the equation for $\dot{I}_a$ is multiplied by $NCF(I_b)$ (although unlike PF below, it is possible that $a = b$). Thus, NCF causes an intention to increase activation on itself more strongly and decreases the impact of negative activation from conflicting intentions. Figure 5.2 illustrates the effects of the non-conflict factor for agents with varying initial I_{ML} magnitudes. I_{ML} is the maximal intention for each agent, but its magnitude (and the magnitudes of other intentions) is higher for high-numbered agents. When compared with the baseline behavior (that is, behavior without NCF), when the maximal activation is very high (e.g., for agent 9), non-conflict factor causes that activation to remain very high due to lack of conflict from other intentions; when the maximal intention is lower, conflict with other intentions causes that maximal intention to be eventually resequenced (illustrated in the figure by the "bump" downward in activation trends) and then, once no longer maximal, to be decreased more rapidly than baseline.

5.3.2. Intentions resist reconsideration. Reconsideration resistance for intention I diminishes (negative) impacts of other intentions on I, as explained in section 3.3.1. Unlike non-conflict factor NCF, which changes I's influence on every other intention, persistence factor PF alters the influence of other intentions on I.

As indicated in equation 12, PF acts on the terms from other intentions in the differential equation for the intention of interest. For example, for the differential equation for I_{ML}, $PF \leq 1$ is multiplied by the term for every other intention; therefore, the conflicting intention has less effect on I_{ML} if I_{ML} is greater. In experiments of PF, intentions

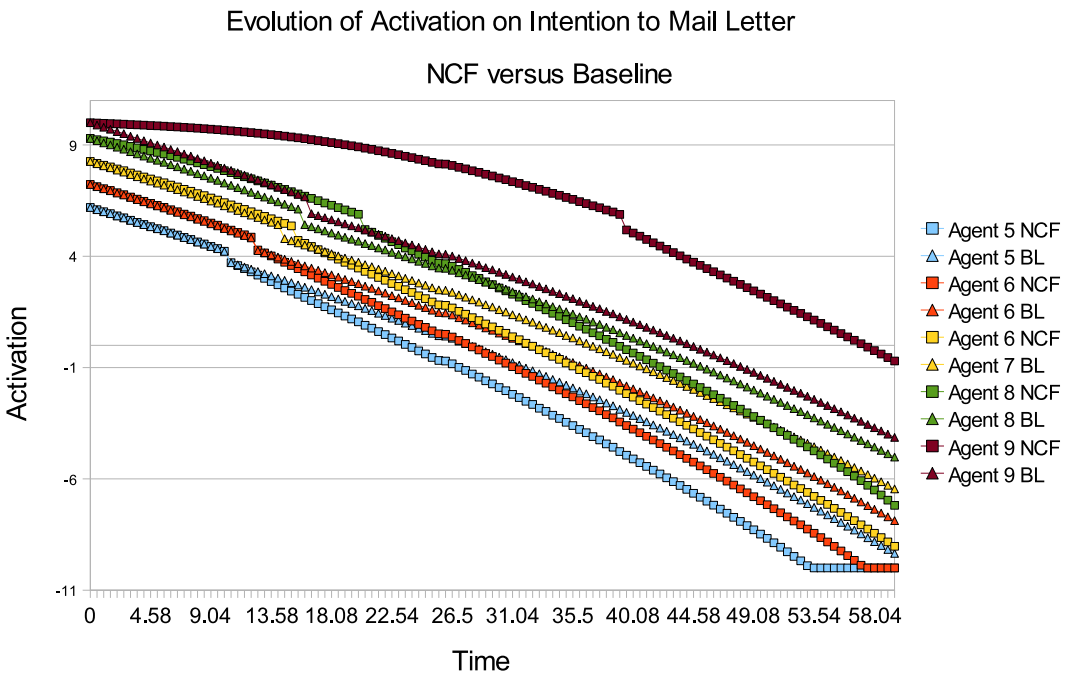

FIGURE 5.2. The effects of non-conflict factor on evolution of activation on I_{ML}. Highly activated intentions push conflicting intentions down, and thus themselves remain highly activated; once resequenced, previously maximal intentions decrease more rapidly than baseline intentions because other, more highly activated intentions (not shown) cause their activation to decrease with NCF.

with higher activations decreased more slowly than intentions with lower activations when PF was present, although the effects are subtle and not shown here.

5.3.3. Intentions control conduct. As in Bratman's definition [**17**], intentions are afforded special status over desires in HDCAs: an agent's strongest intentions need not correspond to its strongest desires; and intentions, not desires, serve to control behavior. These features are demonstrated in a simulation similar to the one described for section 5.1 above. One notable difference is that in the conduct control simulation, an agent begins with the belief that every task is incomplete, so it will not dynamically

	ML	GC	DC	HH	GB/BoB	BuB
Desires	3	9	8	2	2	—
Intentions	5	4	4	2	2	-1

TABLE 5.3. Initial intention and desire activations for learning demonstration

re-sequence I_{ML} to avoid mailing the letter. The movie for this simulation is available on the web at [**4**].

In this simulation, the agent begins navigating toward the school in `GetChild` mode, but quickly finds that its intended path is blockaded. As a result, it deliberatively re-plans its intended task sequence, although I_{GC} remains maximal, and finds a new path to the school. The agent in this simulation also has a rule to do exactly one of `BuyBook` and `BorrowBook`, so during deliberation, I_{BuB} is assigned a highly negative activation because the agent has a stronger activation on I_{BoB}, and therefore plans to borrow, not buy, a book. Effects of this re-planning can be observed in the circles illustrating intention, which discretely change size. In this simulation, desires have very different activations than intentions, and it is clear that intentions guide behavior. Additionally, when the ordering on intentions changes—in response to deliberation or because of natural cognitive evolutions—the ordering of desires does not change, and vice versa. It is apparent, both by design and in practice, that intentions control HDCA conduct.

5.4. Hebbian Learning

As described in Chapter 4, HDCAs can learn to behave reactively based on information about their environments that would typically be handled deliberatively. The Hebbian learning demonstration described in this section is a comparison of two agents: one has learned to account for target distances through Hebbian learning, and another has not completed any learning. In this demonstration, agents navigate a grid city environment identical to the one in Figure 5.1, with initial intentions and desires as indicated in Table 5.3.

To test Hebbian learning, one agent A_H (for *Hebbian*) was trained in the grid-city environment using equation 16. Training involved the agent following a pre-specified path around the environment that took it to within a radius of perception (the distance of one block and one street) of every target, without completing any actions (i.e., without transitioning between modes). At every timestep, intention coefficients were calculated for actions with activated task targets, as described in section 4.1 and equation 16, with a value of $c_1 = 3 \times 10^3$ in the equation.

A_H was then tested against another agent A_{NH} (for *non-Hebbian*); the two were identical except that A_{NH} had the coefficients of A_H before training. Both agents started at the same position with the same BDI activations. (This starting location and set of activations was also identical to those during the training phase for A_H). During the course of the run, the agents first follow similar paths when activations are high on intentions for actions that are far away from other actions, such as `MailLetter`; when activations increase for actions close to other actions, however—such as when agents have high activation on intentions to `DepositCheck`—the agents' behavior diverges and A_H completes actions in order of distance while A_{NH} does not. This demonstration shows that distance information can be encoded on the reactive level using Hebbian learning.

To verify that learning did not affect the way intentions are used, properties of intention were tested similarly to the methods described in section 5.3. Results are similar to those attained from the previous experiments, and are not repeated here.

CHAPTER 6

Conclusion

Hybrid dynamical cognitive agents (*HDCAs*) are hybrid reactive/deliberative agents that have *dynamical intentions* to guide goal-directed behavior. Representations of HDCAs' intentions are shared by both reactive and deliberative levels, cleanly integrating simple deliberation with dynamical reactive behavior. HDCAs' decision making is based on the Belief-Desire-Intention model; in the current framework, beliefs, desires and intentions have continuously varying activation values that indicate salience to the agent. BDI elements are connected in a spreading activation network, and activations on BDI elements evolve continuously as a result of spreading activation from other elements. A type of neuronally-inspired learning called Hebbian learning alters spreading activation connections in order to reduce the burden of deliberation; Hebbian learning trains HDCAs' reactive systems to handle information, such as distances between targets, that is typically processed by deliberative systems. In this thesis, demonstrations of HDCAs in animated grid-city environments illustrate successful goal-directed behavior with replanning on both reactive and deliberative levels, as well as the effects of learning.

6.1. Discussion and Future Work

Proponents of reactive agent architectures [**14, 19**] characterize deliberation as a slow, computationally expensive process that risks reasoning with outdated information in dynamic environments. In the HDCA model described by this thesis, deliberation occurs instantaneously on transitions between modes, and therefore does not represent the time that a real-world agent (human or robot) might take for deliberation. This might be addressed by including an additional `Deliberate` mode that agents enter whenever deliberation is invoked (as described in section 3.2); this mode would slow

or freeze agents' navigation behavior to model the mental load of deliberation. Time spent in `Deliberate` mode could be moderated by cognitive intensity of the deliberation problem as measured, for instance, by the number of actions being considered for a new plan. For a more ecologically valid model, data about the amount of time humans take for certain deliberations might inform the amount of time HDCAs spend in `Deliberate` mode. This type of mode extension is simple in the modular framework described, and would not require significant changes to existing modes or behaviors.

Deliberation is invoked when agents must select from between two intentions with similar activation values; the current threshold for similarity is activation difference of less than two. This threshold could be adjusted to alter the frequency of deliberation or could be made dynamic, based on the agent's cognitive state. For example, threshold value could decrease if many BDI elements are highly active, indicating that the agent has a large "mental load" (because many beliefs, desires, and intentions are salient) and does not have as many resources for deliberation. Greater mental load might lead to less deliberation (i.e., the threshold for deliberation is decreased) or slower deliberation.

Dynamic representations of cognitive and physical elements—such as desires and intentions as well as heading angle and position—allow for arbitrarily precise measurements of system state. This arbitrary precision might enable interesting behavioral dynamics. Specifically, the amount of time taken before a task is completed, or certain perceptions along the way, might have meaningful impacts on how and when intentions are resequenced and therefore on future behavior. For instance, in a multi-agent simulation, agents with similar intentions but different initial locations, and therefore different paths to complete tasks, might display different behaviors resulting from initially minute differences in cognitive state. Current research in progress [**3**] suggests that in the child's game of Tag, small differences in when agents get tagged can affect whether or not those agents get tagged again many timesteps later. Arbitrarily precise measurements of state are underutilized in the current model, but might lead to interesting, complex behavior in HDCAs.

Similarly, a more dynamic environment might highlight benefits of dynamical cognitive representations. Agents in a rapidly changing environment have less time to deliberate, because information used for deliberation might change while agents are reasoning; agents that can learn to transfer deliberative-level behaviors like the distance bias to reactive systems might outperform agents that must deliberate about such behaviors. The current simulation domain offers some dynamic interaction from a moving blockade, which changes position at arbitrary times and restricts agent access to certain streets. A more dynamic environment, potentially one with interactions among agents, might further demonstrate the strengths of dynamic agent architectures.

The particular implementation of Hebbian learning applied in this simulation takes advantage of the fact that proximate targets can be perceived simultaneously, thereby enabling ground concepts to be simultaneously co-activated. Other potentially learned biases or rules, however, might not be so conveniently arranged. Indeed, part of Hebb's ideas about learning [**33**] included the observation that learning often occurs more quickly than neuronal growth can occur—neurons do not develop additional dendrites, or "knobs" as he referred to them, as rapidly as memories seem to be formed. To address this, Hebb postulated a feedback loop that maintained some activation in a cycle of neurons after the initial stimulus disappeared. Memories persist in this feedback loop until biological growth can permanently encode them. Such a feedback loop might be useful for HDCA learning in cases where elements are not immediately co-active. A feedback loop might make it possible to keep one intention or ground concept "in mind" until a second is encountered, at which time the connection between these elements can be strengthened; similarly, it can be used to strengthen connections between elements well after those elements have stopped being perceived.

The HDCA model presented here is a basic framework; extensions that improve dynamicism, learning, and interactions could take advantage of several features of the model. Currently, for instance, individual agents do not interact in any way with other agents in their environment. Implementations that allow for agent communication, such

as sharing information by instantaneously altering other agents' beliefs, might allow for cooperation among agents. Such communication could make agents more efficient, for instance, by having them share information about positions of blockaded streets, and might allow agents to perform tasks that require involvement from multiple agents.

Potential extensions of the HDCA framework could also take advantage of HDCAs' dynamicism by implementing these agents in highly stochastic and dynamic environments, where actions have uncertain effects and agents must reorganize plans quickly. Robots navigating around an office environment, for instance, provide a rich dynamic domain; these agents must handle unexpected obstacles, blocked paths, and a variety of navigation-based tasks.

HDCAs might be applied wherever real-time agents are required. For instance, HDCAs might be background characters in animated video games, where each agent's individual beliefs, desires, and intentions can lead to unique but realistic behavior. Individual HDCAs might function well as office assistant robots, with simple actions (such as `DeliverItem`) and dynamical navigation for obstacle avoidance. Potential non-agent implementations include systems that build models of human behavior, such as patient monitoring systems that can alert a caretaker when patient behavior deviates from predictable norms.

Bibliography

[1] Eric Aaron. Personal communication, 2007–2009.

[2] Eric Aaron and Henny Admoni. A framework for dynamical intention in hybrid navigating agents. In *Proceedings of the Fourth International Conference on Hybrid Artificial Intelligence Systems*, 2009. To appear.

[3] Eric Aaron and Henny Admoni. Hybrid dynamical cognitive agents for animations and video games. In preparation, 2009.

[4] Eric Aaron and Henny Admoni. Supplementary material for HAIS '09. http://eaaron.web.wesleyan.edu/hais09_supplement.html, 2009.

[5] Eric Aaron, Franjo Ivančić, and Dimitris Metaxas. Hybrid system models of navigation strategies for games and animations. In Claire J. Tomlin and Mark R. Greenstreet, editors, *Hybrid Systems: Computation and Control*, volume 2289 of *Lecture Notes in Computer Science*, pages 7–20. Springer-Verlag, 2002.

[6] Eric Aaron, Harold Sun, Franjo Ivančić, and Dimitris Metaxas. A hybrid dynamical systems approach to intelligent low-level navigation. In *Proceedings of Computer Animation*, pages 154–163, 2002.

[7] Henny Admoni. Decision making and learning for hybrid dynamical agents. Honors thesis, Wesleyan University, 2008.

[8] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 268–272, 1987.

[9] Rajeev Alur, Calin Belta, Franjo Ivančić, Vijay Kumar, Max Mintz, George J. Pappas, Harvey Rubin, and Jonathan Schug. Hybrid modeling and simulation of biomolecular networks. In *Hybrid Systems: Computation and Control*, volume 2043 of *Lecture Notes in Computer Science*, pages 19–32. Springer-Verlag, April 2001.

[10] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A. Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, February 1995.

[11] Rajeev Alur, Thomas A. Henzinger, Gerardo Lafferriere, and George J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.

[12] John R. Anderson. A spreading activation theory of memory. *Journal of Verbal Learning and Verbal Behavior*, 22(3):261–295, June 1983.

[13] Ronald C. Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6(1–2):105–122, 1990.

[14] Ronald C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, USA, 1998.

[15] Aude Billard and Gillian Hayes. DRAMA, a connectionist architecture for control and learning in autonomous robots. *Adaptive Behavior*, 7(1):35–63, 1999.

[16] James E. Black and William T. Greenough. Developmental approaches to the memory process. In Joe Martinez and Raymond Kesner, editors, *Neurobiology of Learning and Memory*, pages 55–88. Academic Press, 1998.

[17] Michael Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, Cambridge, MA, USA, 1987.

[18] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[19] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[20] Rodney A. Brooks. Intelligence without reason. Technical Report 1293, Massachusetts Institute of Technology Artificial Intelligence Laboratory, April 1991.

[21] Jason R. Chen and Brenan J. McCarragher. Programming by demonstration - constructing task level plans in a hybrid dynamic framework. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1402–1407, San Francisco, CA, April 2000.

[22] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 1990.

[23] Allan M. Collins and Elizabeth F. Loftus. A spreading-activation theory of semantic processing. *Psychological Review*, 82(6):407–428, 1975.

[24] Mary L. Courage and Mark L. Howe. From infant to child: The dynamics of cognitive change in the second year of life. *Psychological Bulletin*, 128(2):250–277, 2002.

[25] Lavindra de Silva and Lin Padgham. A comparison of BDI based real-time reasoning and HTN based planning. In G. I. Webb and Xinghuo Yu, editors, *AI 2004: Advances in Artificial Intelligence*, volume 3339 of *Lecture Notes in Computer Science*, pages 1167–1173. Springer-Verlag, 2004.

[26] Lavindra de Silva and Lin Padgham. Planning on demand in BDI systems. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Monterey, CA, USA, 2005. (Poster).

[27] Magnus Egerstedt. Behavior based robotics using hybrid automata. In N. Lynch and B. Krogh, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *Lecture Notes in Computer Science*, pages 103–116. Springer-Verlag, 2000.

[28] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence 2*, pages 189–208, 1971.

[29] Jerry A. Fodor. *The Modularity of Mind: An Essay on the Faculty Psychology*. MIT Press, Cambridge, MA, USA, 1983.

[30] Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, 1987.

[31] Lakshmi J. Gogate, Arlene S. Walker-Andrews, and Lorraine E. Bahrick. The intersensory origins of word comprehension: An ecological-dynamic systems view. *Developmental Science*, 4(1):1–37, 2001.

[32] Siome Goldenstein, Menelaos Karavelas, Dimitris Metaxas, Leonidas Guibas, Eric Aaron, and Ambarish Goswami. Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computers & Graphics*, 25(6):983–998, December 2001.

[33] Donald O. Hebb. *The Organization of Behavior*. John Wiley & Sons, Inc., New York, NY, USA, 1949.

[34] Clinton Heinze, Simon Gross, and Adrian Pearce. Plan recognition in military simulation: Incorporating machine learning with intelligent agents. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, Workshop on Team Behavior and Plan Recognition*, pages 53–63, Stockholm, Sweden, 1999.

[35] Thomas A. Henzinger. The theory of hybrid automata. *11th Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 278–308, 1996.

[36] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1–2):110–122, December 1997.

[37] François F. Ingrand, Michael P. Georgeff, and Anand S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, 1992.

[38] Patricia A. Jaques and Rosa M. Vicari. A BDI approach to infer student's emotions in an intelligent learning environment. *Computers & Education*, 49:360–384, 2007.

[39] Hiroaki Kawashima. *Interval-Based Hybrid Dynamical System for Modeling Dynamic Events and Structures*. PhD thesis, Kyoto University, 2007.

[40] Pattie Maes. Situated agents can have goals. In *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 49–70. MIT Press, 1991.

[41] Maja J. Matarić. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

[42] James L. McClelland, David E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2. MIT Press, Cambridge, MA, USA, 1986.

[43] Jörg P. Müller, Markus Pischel, and Michael Thiel. Modeling reactive behaviour in vertically layered agent architectures. In Michael J. Wooldridge and Nicholas R. Jennings, editors, *Intelligent Agents: Theories, Architectures, and Language*, volume 890 of *Lecture Notes in Artificial Intelligence*, pages 261–276. Springer-Verlag, 1995.

[44] Nils J. Nilsson. Shakey the robot. Technical Report 323, SRI International, Menlo Park, CA, USA, April 1984.

[45] Emma Norling. Learning to notice: Adaptive models of human operators. In *Second International Workshop on Learning Agents*, Montreal, Canada, May 2001. ACM.

[46] Emma Norling. Folk psychology for human modelling: Extending the BDI paradigm. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pages 202–209, New York, NY, USA, July 2004. ACM.

[47] David L. Pepyne and Christos G. Cassandras. Optimal control of hybrid systems in manufacturing. *Proceedings of the IEEE*, 88(7):1108–1123, July 2000.

[48] M. Ross Quillian. A revised design for an understanding machine. *Mechanical Translation*, pages 17–29, July 1962.

[49] M. Ross Quillian. Word concepts: A theory and simulation of some basic semantic capabilities. *Behavioral Science*, 12(5):410–430, September 1967.

[50] Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484, 1991.

[51] Anand S. Rao and Michael P. Georgeff. An abstract architecture for rational agents. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 439–449, 1992.

[52] Anand S. Rao and Michael P. Georgeff. BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS '95)*, pages 312–319, San Francisco, CA, USA, June 1995.

[53] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[54] David E. Rumelhart, James L. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA, USA, 1986.

[55] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

[56] Larissa K. Samuelson and Jessica S. Horst. Confronting complexity: Insights from the details of behavior over multiple time scales. *Developmental Science*, 11(2):209–215, 2008.

[57] Sebastian Sardina, Lavindra de Silva, and Lin Padgham. Hierarchical planning in BDI agent programming languages: A formal approach. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1001–1008, Hakodate, Japan, 2006. ACM.

[58] Wei Shao and Demetri Terzopoulos. Autonomous pedestrians. *Graphical Models*, 69:246–274, September–November 2007. Special issue on SCA 2005.

[59] Lokendra Shastri. Advances in SHRUTI—A neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence*, 11:79–108, 1999.

[60] Linda B. Smith and Larissa K. Samuelson. Different is good: Connectionism and dynamic systems theory are complementary emergentist approaches to development. *Developmental Science*, 6(4):434–439, September 2003.

[61] Linda B. Smith and Esther Thelen. Development as a dynamic system. *TRENDS in Cognitive Sciences*, 7(8):343–348, August 2003.

[62] Michael Spivey. *The Continuity of Mind*. Oxford Psychology Series. Oxford University Press, New York, NY, USA, 2007.

[63] Esther Thelen and Linda B. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action*. MIT Press/Bradford Book Series in Cognitive Psychology. MIT Press, Cambridge, MA, USA, 1994.

[64] Esther Thelen and Linda B. Smith. Dynamic systems theories. In William Damon (series ed.) and Richard M. Lerner (volume ed.), editors, *Handbook of Child Psychology: Theoretical Models of Human Development*, volume 1, chapter 6, pages 563–633. Wiley, New York, NY, USA, 5th edition, 1998.

[65] Claire Tomlin, George J. Pappas, and Shankar Sastry. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, April 1998.

[66] Arjan van der Schaft and Hans Schumacher. *An Introduction to Hybrid Dynamical Systems*, volume 251 of *Lecture Notes in Control and Information Sciences*. Springer, 2000.

[67] Andrzej Walczak, Lars Braubach, Alexander Pokahr, and Winifried Lamersdorf. Augmenting BDI agents with deliberative planning techniques. In R. H. Bordini et al., editor, *Programming Multi-Agent Systems*, volume 4411 of *Lecture Notes in Computer Science*, pages 113–127. Springer-Verlag, 2007.

[68] Dave Wooden, Matt Powers, Magnus Egerstedt, Henrik Christensen, and Tucker Balch. A modular, hybrid system architecture for autonomous, urban driving. *Journal of Aerospace Computing, Information, and Communication*, 4:1047–1058, December 2007.

APPENDIX A

Sample Code

This appendix contains MATLAB code from HDCA implementations described in this paper. Sections of code that have been omitted for clarity are labeled. Some lines of code are annotated in the margins; these notes refer to equations in the main text that are implemented here.

A.1. Main Simulation Loop

```
% This is the main simulation loop.  It builds the grid-city
% environment and initializes agents, then iterates through
% the list of agents, updating them 1 timestep in the correct
% mode using a switch statement.  This loop continues until
% all agents have finished moving (i.e., all agents are at
% target Home).

function mainLoop

% initialize simulation clock
global time;
global timestep;
global endtime;

% initialize mode names as constants
global MAILLETTER;
global PICKUPCHILD1;
global DEPOSITCHECK;
global HURRYHOME;
global BORROWBOOK;
global BUYBOOK;
global STAND;

% code deleted here
% ...

% create grid world
numOfBlocks = 4;
street = 12; % width of a street
block = 8; % width of a block, i.e., diameter of a building
```

```
worldsize = numOfBlocks * (block + street);

% code deleted here
% ...

% initialize agents
numAg = 1; % potential for multi-agent simulation
for i = 1:numAg
    agt = initAgent(i);
    agTemp(i) = agt;
end
ag = agTemp;

numAgtsMoving = numAg;

% code deleted here
% ...

while (numAgtsMoving ~= 0)

    % each agent performs appropriate action
    for i = 1:numAg

        x = ag(i).posn(1);
        y = ag(i).posn(2);
        oldMode = ag(i).mode;

        oldBeliefs = ag(i).beliefs;
        oldIntentions = ag(i).intentions;
        oldDesires = ag(i).desires;

        switch oldMode
            case MAILLETTER
                ag(i) = mailletter(ag(i));
            case PICKUPCHILD1
                ag(i) = pickupchild1(ag(i));
            case DEPOSITCHECK
                ag(i) = depositcheck(ag(i));
            case HURRYHOME
                ag(i) = hurryhome(ag(i));
            case BORROWBOOK
                ag(i) = borrowbook(ag(i));
            case BUYBOOK
                ag(i) = buybook(ag(i));
            case STAND
```

```
                ag(i) = stand(ag(i));
            otherwise
                error('unknown mode in ag(i) switch');
        end

        newMode = ag(i).mode;
        newBeliefs = ag(i).beliefs;
        newDesires = ag(i).desires;
        newIntentions = ag(i).intentions;

        if newMode == STAND
           numAgtsMoving = numAgtsMoving - 1;
        end
    end

    time = time + timestep;
end

% code deleted here
% ...

-----------------------------------------------
% Initialize an agent

function agt = initAgent(agtnumber)

% code deleted here
% ...

agt.name = num2str(agtnumber);
agt.num = agtnumber;
agt.size = 0.005*worldsize;

% code deleted here
% ...

agt.posn = [1 34];

agt.beliefs(letterMailed) = 10;
agt.beliefs(c1Here) = -10;
agt.beliefs(checkDepd) = -10;
agt.beliefs(haveCheck) = 10;
agt.beliefs(atHome) = -10;
agt.beliefs(bookBorrowed) = -10;
agt.beliefs(bookBought) = -10;
```

```
agt.desires = [3 9 8 2 1];
agt.intentions = [6 9.3 10 1 5 -1];

% code deleted here
% ...
```

A.2. Example Mode (Deposit Check Function)

```
% DepositCheck mode.  First, guards are evaluated; if
% a guard is tripped, that transition is taken.  If no
% guards are tripped, function checks whether the agent
% is at its target so that it can complete the task.  If
% so, certain beliefs, desires and intentions are set
% discretely; guards will be tripped on the next cycle
% through this mode.  If agent is not at target, update
% desires and intentions, then update physical location in
% grid city.

function agent = depositcheck(agent)

% code deleted here
% ...

% Evaluate guards

% returns two most highly-active intentions
[tempsi,secondsi] = toptwomaxseqints(agent);
if tempsi ~= DEPOSITCHECK
    % transition to new mode, either tempsi or secondsi
    % depending on the results of deliberation
    agent = takeTransition(agent,tempsi,secondsi);
else
    atTarg = isNearDistance(agent, ...
        targets(bank).loc,(targsize/2+agent.size));

    if (atTarg && agent.beliefs(haveCheck) ~= -10)
        agent.beliefs(haveCheck) = -10;
        agent.beliefs(checkDepd) = 10;
        agent.desires(depCheck) = -10;
        agent.intentions(bank) = -10;
    else %update a step, prepare to be in this mode again

        % code deleted here
        % ...
```

```
        % Update BDI based on differential equations
        agent = updateDesires(agent, DEPOSITCHECK);
        agent = updateIntentions(agent,DEPOSITCHECK);

        % Update phys. locat'n based on dynamical navigation

        % code deleted here
        % ...
    end

    % code deleted here
    % ...

end
```

A.3. Intention Update Function

```
% Returns the updated agent with recalculated intentions
% based on the mode.
function agent = updateIntentions(agent,mode)

% code deleted here
% ...

% nonconflict factor array
ncfarray = (power(agent.intentions,8)/(power(10,8)/(1.6)) + ...
   power(agent.intentions,9)/(power(10,9)/(0.8)));

% persistence factor denominator (1e-12 avoids div by 0)
perfactdenom = sum(abs(agent.intentions)) + 1e-12;
% persistence factor for I_DC
perfactBK = (abs(agent.intentions(bank))) / perfactdenom;

% code deleted here
% ...

% update intention I_DC
h = agent.icoeffs(3,:,mode); % array of integers between 0 and 10
dI(bank) = h(1) * agent.beliefs(haveCheck) - ...
    h(2) * agent.beliefs(checkDepd) + ...
    h(3) * agent.desires(depCheck) - ...
    h(4) * agent.desires(getc1) - ...
    h(5) * agent.desires(mailLetter) - ...
    h(6) * agent.desires(goHome) - ...
    h(7) * agent.desires(getBook) + ...
    h(8) * (1+ncfarray(bank)) * agent.intentions(bank) - ...
    h(9) * (1-perfactBK)*(1+ncfarray(home)) * ...
```

Eq. 15

Eq. 13

Eq. 4 and Eq. 9

```
     agent.intentions(home) - ...
    h(10) * (1-perfactBK)*(1+ncfarray(school1)) * ...
     agent.intentions(school1) - ...
    h(11) * (1-perfactBK)*(1+ncfarray(postoffice)) * ...
     agent.intentions(postoffice) - ...
    h(12) * (1-perfactBK)*(1+ncfarray(library)) * ...
     agent.intentions(library) - ...
    h(13) * (1-perfactBK)*(1+ncfarray(bookstore)) * ...
     agent.intentions(bookstore) - ...
    h(14) * distance(x,y, ...
     targets(bank).loc(1),targets(bank).loc(2));

% code deleted here
% ...

agent.intentions = ...
    max(-10,min(10, agent.intentions + dI * timestep));
```

Eq. 5

A.4. Deliberation Function

```
% Agent deliberates about what action to perform using intention
% activations, distances to targets, and domain-specific
% knowledge (elements of domain-specific knowledge are elided
% for these excerpts).

function agent = deliberateToNewMode(agent)

% code deleted here
% ...

% construct the plan by selecting tasks in order
while ~isempty(temptasklist) % while there are still tasks to plan
    maxscore = -10*scalingfactor - worldsize; % min possible score
    maxtask = 0;
    maxtaskposn = 0;
    for t = 1:length(temptasklist)
        task = temptasklist(t);
        % calculate Manhattan distance to target
        dist = abs(targets(task).loc(1) - prevloc(1)) + ...
            abs(targets(task).loc(2) - prevloc(2));
        % compute score based on distance and intention activation
        score = agent.intentions(task)*scalingfactor - dist;
        if score > maxscore
            maxscore = score;
            maxtask = task;
            maxtaskposn = t; % position of maxtask in temptasklist
        end
```

Eq. 10

```
    end
    plannedtasklist = [plannedtasklist maxtask];
    prevloc = targets(maxtask).loc;
    temptasklist(maxtaskposn) = []; % remove task from considerat'n
end

% code deleted here
% ...

% reset agent's intentions to correspond to the new plan
maxactiv = 8; % highest activat'n value for first intention in plan
for j = 1:length(plannedtasklist)
    nt = plannedtasklist(j);
    agent.intentions(nt) = maxactiv / j;
end

% code deleted here
% ...
```

Eq. 11

A.5. Hebbian Learning Training Function

```
% This function trains the agent on the environment.  If two
% locations are simultaneously within an agent's radius of
% perception, these locations are said to be "co-active" and
% activation links between them are strengthened.  Distance
% threshold increases as the agent spends more time in its
% environment.

function icoeffs = train(agent)

% code deleted here
% ...

while ~isNearDistance(agent, lasttargpos, agent.size*2.1)
    x=agent.posn(1);
    y=agent.posn(2);

    % code deleted here
    % ...

    % observe which targets are within distance threshold
    for t=1:6 % length of target list
        tarloc=targets(t).loc;
        if isNearDistance(agent,tarloc,distthresh)
            if locgcs(t,2)==0 % if target is seen for first time
                locgcs(t)=(1.5 * locgcs(t,1)) + 2;
```

```
                locgcs(t,2)=1; % mark target as being seen
            end
        else
            if locgcs(t,2)==1 %target was seen, now out of range
                locgcs(t,2)=0; %mark targ as out of viewing range
            end
            locgcs(t,1) = max(0,locgcs(t,1) + dLocTar*timestep);
        end
    end

    % code deleted here
    % ...

    % create a list of nearby targets
    nearbytargs = [];
    for b = 1:length(locgcs)
        if locgcs(b,1) > 0.001
            nearbytargs = [nearbytargs b];
        end
    end

    for n = 1:length(nearbytargs)
        int = nearbytargs(n);

        % for every mode, weaken inhibitory connections b/w this
        % intention and every other intention in nearbytargs list
        for ot = 1:length(nearbytargs)
            otherint = nearbytargs(ot);
            if otherint ~= int

                % code deleted here
                % ...

                cfm = (-1/3)*locgcs(otherint,1)/coeffDivisor;
                icoeffs(int,otherint+offset,:) = ...
                    icoeffs(int,otherint+offset,:)+cfm*timestep;
            end
        end
    end

% code deleted here
% ...
```

Eq. 16