AN INFRASTRUCTURE FOR SELF-MOTIVATED, CONTINUALLY PLANNING AGENTS IN VIRTUAL WORLDS

by

Daphne H. Liu and Lenhart Schubert

Technical Report TR-2012-985
in the Department
of
Computer Science
Arts, Science & Engineering
Edmund A. Hajim School of Engineering & Applied Sciences

University of Rochester
December 2012

Abstract

We present a flexible initial framework for defining self-motivated, self-aware agents in simulated worlds, planning continuously so as to maximize long-term rewards. While such agents employ reasoned exploration of feasible sequences of actions and corresponding states, they also behave opportunistically and recover from failure, thanks to their continual plan updates and quest for rewards. Our framework allows for both specific and general (quantified) knowledge, and for epistemic predicates such as knowing-that and knowing-whether. Because realistic agents have only partial knowledge of their world, the reasoning of the proposed agents uses a weakened closed world assumption; this has consequences for epistemic reasoning, in particular introspection. The planning operators allow for quantitative, gradual change and side effects such as the passage of time, changes in distances and rewards, and language production, using a uniform procedural attachment method. Question answering (involving introspection) and experimental runs are shown for our particular agent ME in a simple world, demonstrating the value of continual deliberate, reward-driven planning. Though the primary merit of agents definable in our framework is that they combine all of the above features, they can also be configured as single or multiple goal-seeking agents, and as such perform comparably with some recent experimental agents.

Keywords: Communicative agents; Continual planning; Deliberate and opportunistic behavior; Incomplete knowledge; Introspection; Self-aware agents; Self-motivated cognitive agents.

Acknowledgments

This work has been supported by NSF grants IIS-0916599 and IIS-1016735.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the long term, we are interested in creating linguistically competent, intelligent, human-like agents. Such agents will have to be able to converse, plan flexibly in an incompletely known world, and make inferences over a substantial knowledge base, all while pursuing their own rewards. Though many AI systems have demonstrated *some* of these abilities, none, to our knowledge, have integrated them all. We present a simple initial framework and platform (see also [49, 50]) for defining such self-motivated cognitive agents in simulated "gridworlds" (worlds that may contain various entities at various accessible places connected by paths), and exemplify this framework with an agent dubbed *ME* (pronounced "em-ee" for Motivated Explorer). ME can plan and think ahead (where actions can take up time and produce gradual change), deal with unforeseen environmental events, make inferences about states (including its own and other agents' knowledge and wants), engage in limited question-answering with the user and other agents, and do so indefinitely and autonomously, driven by the expected cumulative rewards or costs of its contemplated sequences of actions and anticipated future states.

Unlike traditional planning agents, which are constrained to act and react in accordance with user-specified goals, constraints, and preferences, agents defined in our framework are self-motivated. In that respect they resemble policy-driven agents, but at the same time they are *cognitive* agents, able to plan ahead – sufficiently well not only to "thrive" in their worlds, but also, with some minor tweaks, to solve classical STRIPS-like planning problems and more recent continual planning problems in incompletely known worlds, as will be seen in Chapter 7), to entertain beliefs about themselves and other agents, and to engage in simple question-answering dialogues.

In Chapter 2, we give a general outline of what we mean by a self-motivated cognitive agent, and of our development framework for implementing such agents. In Chapter 3 we provide some details of how actions are defined for an agent, the planning process, and the somewhat altered syntax and meaning of action definitions when we are specifying "actual" actions (as carried out when "running" the simulated world). In Chapter 4 we turn to the other major aspect of the agent framework, namely the reasoning (including introspection) performed by the agent.

In Chapter 5, we instantiate the agent ME in a simulated world, for the purpose of demonstrating and experimenting with our agent framework. The world is simple, but suffices for illustrating and supporting the points we wish to make. In Section 5.2 we provide a range of examples of question-answering by ME (including questions involving attitudes), and report in Chapter 6 the results of various "runs" of the agent in the simulated world. One set of runs is aimed at demonstrating the opportunistic, future-directed behavior of the agent when all its faculties are intact. In another set of runs, the agent is deprived of its ability to plan and reason ahead (from another perspective, of its self-awareness), and this leads to the expected deterioration in its behavior. In a third experiment, the agent is rendered doggedly goal-directed (towards one specific consumption goal) by upward adjustment of the rewards it subjectively assigns to the chosen goal and to types of actions and states that lead towards it, while zeroing-out most other reward estimates. This shows that such goal-directedness will fail to exploit available opportunities, and greatly reduce net utility. In Chapter 7, we explore the applicability of our agent framework to classical and continual goal-directed planning, and in Chapter 8 we discuss related work on behavioral and game agents, and agents that reason and converse. We conclude with a summary of our work and discuss future directions.

Chapter 2

Self-Motivated Cognitive Agent Framework

We conceive of a self-motivated cognitive agent as one whose activity is governed by never-ending planning and self-aware reasoning aimed at optimizing long-term, cumulative rewards derived from the actions taken and states reached (as previously suggested in [58, 49]). We think of these rewards (or penalties) as subjective, much as human beings experience pleasure from eating, companionship, sex, discovery, acquisition of goods, pastimes, etc., and distress from fatigue, injury, loss of possessions, etc. Thus a self-motivated agent, in our sense, does not pursue goals for their own sake, but rather for their anticipated contribution to the sum of agent's subjective rewards over the long term. [35] and similarly [55], in discussing self-motivation, focus on the need for explicitly represented goals that can be created, reasoned about, chosen from, and translated into behavior. While our framework provides these functions, we feel that the genesis of goals via projected cumulative rewards and costs, and a model of self – i.e., self-awareness – are also essential ingredients.

An agent is not assumed to have complete knowledge about the world, either about the current state or about exogenous events, but it should be able to observe new or altered facts as it goes. By its nature, such an agent will manifest *opportunistic*, or *situated* behavior; i.e., as new facts become apparent to it, or actions fail, its continual planning will take these into account so as to exploit newly recognized opportunities or avoid newly recognized threats.

Accordingly, the framework and (Lisp-based) platform we have built treat planning as continual construction, evaluation, and partial execution of sequences of potential actions. The evaluation assumes that both actions and states can be intrinsically rewarding or odious, where the extent of such utility or disutility can depend on the exact action parameters involved and on arbitrary descriptive numerical features of the world and the agent. Therefore, the design of the Lisp functions that measure action and state utility is left open to the user. Actions and states with large anticipated utility will tend to be favored by the planning process, which is designed to add up total utilities over courses of action, and propagate these back to allow a current choice of action that is seemingly optimal "in the long run". Rewards and costs need not be limited to actions and states that are intrinsically rewarding or odious, but may perfectly well be associated with actions or states that tend to foreshadow good or bad outcomes. One can imagine adding a learning component to this scheme, where only intrinsically pleasant or unpleasant actions or states initially have non-null utilities, while other actions and states gain or lose utility because experience indicates that they lead to favorable or unfavorable consequences. However, our most immediate focus has been on basic cognitive equipment, not on learning.

To qualify as a *cognitive* system, an agent should be able to plan and reason with an expressively rich language. We might have chosen a situation calculus-based framework like Golog (e.g., [67]), but this would have limited the complexity of the agents and worlds that could be practically implemented. Instead, we chose a STRIPS-like representation of actions and states, with allowance for quantitative preconditions and effects (including variable elapsed time), handled by a very simple, general procedural attachment syntax. Moreover, the logic of state descriptions allows for propositional attitudes such as knowing-that, knowing-whether, and wanting. This is essential for formalizing knowledge-acquisition actions, such as asking a question, and for answering questions about the agent's own attitudes. Details of the action representation and planning process are deferred to the next chapter.

As mentioned in the Introduction, the current framework places agent ME and other objects (and possibly other agents) in a simulated "gridworld" of named points (locations) and connections (roads). The primary functions for defining a world are the following:

```
(def-roadmap points roads)
(def-object obj-type properties)
(place-object name obj-type point associated-things curr-facts propos-attitudes)
```

In *def-roadmap*, the points are arbitrary names and the roads are sequences of type *(road-name point0 dist1 point1 dist2 point2 ...)*. In *def-object*, *obj-type* is a predicate such as *piano* or *sasquatch*, and properties are also predicates, presumed to apply to all objects of that type, such as *is_playable, is_animate*, or *(has_IQ 50). These properties are mapped to implicative clauses such as* `((sasquatch ?x) => (is_animate ?x)).`

The *place-object* function serves to introduce an entity of type *obj-type* into the world, positioning it at *point*. It also allows specification of objects that the new entity "has" (its associated things), arbitrary ground facts about it, and propositionl attitudes (in the case of animate agents) such as *(knows Grunt (that (has ME Banana1)))*. ME is also introduced with the *place-object* function, but the name is recognized as referring to the main actor in the world, planning actions to optimize cumulative rewards.

There is also a general knowledge repository, **general-knowledge**, to which the user can add arbitrary implications with positive antecedents and a positive consequent, with allowance for *knows that* predications and some other epistemic predications; for example,

```
((is_animate ?x) => (can_talk ?x)),
((knows ?x (that ?p)) => ?p).
```

These clauses are used for bounded forward inference in elaborating world state descriptions.

While ME is mobile and capable of exploring the world and interacting with the user and other entities, we generally assume that other entities are stationary and merely reactive in their interactions with ME. (However, in Section 7.3 we adapt our framework to a class of multiagent problems.)

ME's knowledge base is automatically initialized so that it contains the geographical knowledge about the world and the general quantified conditional facts. Its world model is also augmented with specific facts about itself, its initial location, and about the entities at that location. Notably, ME has only partial knowledge about the world, even about entities at its current location. In particular, the designer of ME's world may mark certain predicates as being *occluded* for ME, and these are predicates for local facts that the designer wants to regard as not immediately perceptible to ME. For example, the predicate *is_in* might be occluded, so if ($is\_in\ key1\ box1$) holds, ME does not know this even when standing next to $box1$. Similarly, $knows$ would generally be occluded, so that what another agent at ME's location knows is not apparent to ME. But there are two types of exceptions. First, if a certain argument (usually the subject) of the predicate is ME, then predication is not occluded to ME. For example, self-referential facts like ($has\ ME\ key1$) and (not ($knows\ ME$ ($whether$ ($is\_edible\ fruit3$)))) would be evident to ME (and thus added to ME's world model), despite the general occlusion of *has* and $knows$. Second, ME may find out and remember a fact that would otherwise be occluded, perhaps because it asked a question, read a book containing the fact, or inferred it.

There are two separate views in the gridworld: the simulation ("God's-eye view"), and

ME's knowledge base, which, unlike the former, is very incomplete, containing only what ME can perceive locally (subject to partial occlusion), or has learned through previous perceptions, questioning of others, or inference. The only exceptions are (a) that the gridworld layout is known to ME at initialization, (b) that ME has general knowledge about entity types and their properties, and (c) that ME has knowledge about its own action capabilities. The knowledge gained by ME when it reaches a new location or when the situation changes (e.g., rain, fire) is intended as our model of the ME's perceptions, and it is the only channel through which ME gains knowledge about the existence of particular objects with particular non-occluded properties at particular locations in the gridworld.

The incompleteness of ME's knowledge has several important consequences for the design of the cognitive agent framework. One is that a distinction needs to be made between the simulated world and ME's model of it, in terms of both the effects and the termination conditions of actions (which may be determined by external events, not anticipated by ME). Another is that under conditions of incomplete knowledge, ME cannot use a full closed-world assumption (CWA), and thus must be careful not only in its evaluation of the truth or falsity of action preconditions and effects, but also in its introspections concerning what it knows and does not know. Finally, in such a setting ME is apt to benefit from continual (re-)planning, since every step can bring unanticipated opportunities or hazards.

Chapter 3

Actions, Planning, and Simulation

Operators are defined for ME in a STRIPS-like syntax, with a list of parameters, a set of preconditions, and a set of effects; an action also has an anticipated reward value and an expected duration. As an example, consider the *drink* operator in Figure 3.1 with formal parameters $?h$ for ME's thirst level and $?x$ for the item to be drunk. From ME's perspective, if it is thirsty and possesses a potable item, then ME can drink it for 1 time unit and completely relieve its thirst.

```
(setq drink
   (make-op  :name 'drink  :pars '(?h ?x)
   :preconds '( (is_thirsty_to_degree ME ?h)
                    (> ?h 0.0)
                    (is_potable ?x)
                    (has ME ?x)
                  )
   :effects '( (is_thirsty_to_degree ME 0.0)
                (not (is_thirsty_to_degree ME ?h))
                (not (has ME ?x))
              )
   :time-required 1
   :value '(* 4 ?h)
   )
)
```

Figure 3.1: The *drink* Model Operator

The syntax is essentially that of PDDL [54, 30], the prevailing standard for defining planning domains, especially for systems participating in international planning competitions. We do not make use of conditional effects (which in principle can be dispensed with through use of multiple variants of actions), and quantified ($\forall$ and $\exists$) preconditions and effects. On the other hand, our framework allows for incompletely known worlds, expansion of state descriptions via forward inference, and propositional attitudes of knowing-that, knowing-whether, and wanting, which are essential for a self-aware, communicative agent.

A plan consists of a sequence of actions. An action – an instance of an operator – is created by replacing its formal parameters with actual values obtained by unifying the operator preconditions with the facts in ME's conception of the current state along with the permanently true facts. ME considers taking an action in a given state only if it believes its preconditions to be true according to its model of that state; ME considers such an action *applicable* although it might not elect to perform this action in that state. ME accomplishes planning by first doing a forward search from a given state s, followed by back-propagation to s of the anticipated rewards and costs of the various actions and states reached, to determine a/the seemingly best sequence of actions (i.e., the best plan for securing cumulative rewards). The forward search is bounded by a prespecified search beam, which specifies the number of lookahead levels (the length of the contemplated sequences of actions), the branching factor and the allowable operators at each level. Informed by the projective lookahead, ME will then execute the first action of a/the seemingly best plan and update its knowledge with the action effects and its new observations of non-occluded, local facts. It is then ready to plan forward from the new state.

We wished to make it easy as possible for users to create not only the world's geography, objects, and actions as conceptualized by the main agent, but also to create the "actual" world, i.e., the simulation. We therefore devised a semantically transparent syntax for defining "actual" action types rather similar to ME's models of these actions, but allowing for stepwise updates for multistep actions, and possible interruptions by exogenous events.[1]

Consider the following example of the actual, stepwise version of the *drink* operator in Figure 3.2.

In the simulation, actual actions and exogenous events (also defined with the same syntax) may be tracked in parallel in unit time steps. An active actual action will continue for another time step if and only if none of its stop conditions as given by *stopconds* are true in the current world state. If at least one of them is true in the current world state, then the action will terminate immediately. In either case, the world state will be updated, by computing the effects as given by *deletes* and *adds*, followed by bounded forward inference in conjunction with general world knowledge. At the same time, ME's world model will be updated by computation of ME's observation of the non-occluded, local facts in the new state.

[1]Interested readers can contact the authors to request the gridworld manual, in which syntactic details of operators, the Horn-like knowledge representation, and careful discussions of operational semantics are provided.

```
(setq drink.actual
   (make-op.actual  :name 'drink.actual  :pars '(?h ?x)
   :startconds '( (is_thirsty_to_degree ME ?h)
                  (> ?h 0.0)
                  (has ME ?x)
                  (is_potable ?x)
                )
   :stopconds '( (is_thirsty_to_degree ME 0.0) )
   :deletes '( (is_thirsty_to_degree ME ?#1)
               (has ME ?x)
             )
   :adds '( (is_thirsty_to_degree ME 0.0) )
   )
)
```

Figure 3.2: The *drink* Actual Operator

When an action comes to a stop, whether successfully or unsuccessfully, ME performs its plan search and chooses a step to execute, as already described. This is what makes its planning situated, or opportunistic, or robust in the event of failure – it always bases its choice of the next action on a seemingly best course of action that starts in the situation at hand.

We emphasize that model operators, which reflect ME's understanding of its own actions, are used by ME in its forward projection, whereas the actual operators are used in the simulation. The latter are not planning operators, but in effect state-change operators that reflect what happens if ME performs certain actions, or exogenous events occur. Naturally, the changes wrought by these operators are necessarily "successful" – the gridworld never fails to reflect the changes that actions/events cause. The state of the simulated world is complete (within the confines of the gridworld) whereas the models entertained by ME are not. Furthermore, the only communication channel between the simulated world and ME's model is ME's perception of the simulated world.

Note that preconditions and effects (in both model operators and actual operators) follow a conventional logical syntax of predications and terms, where predications may be negated, and terms may be constants, variables (parameters), or functional expressions (allowing for certain reified propositions or questions, as specified later). Some of the functional terms and predicates may be evaluable by attached procedures – a common technique in reasoning systems. This technique is used in preconditions and effects (and value and duration – reward calculations and elapsed time calculations – if applicable) of both the model actions and the actual actions, and it enables not only quantitative reasoning but also dialogue (e.g., in question-answering). Predicates and functions evaluable by associated procedures include $+$, $-$, $*$, $/$, $<$, $<=$, $>$, $>=$, $=$, *random*, and user-defined predicates or function names ending in ?.

The system will evaluate the evaluable expressions when verifying preconditions and when applying effects (and when contemplating the net utility and duration, if applicable). For example, ($is\_tired\_to\_degree$ ME (+ ?f (∗ 0.5 ($elapsed\_time$?)))), an effect of $walk.actual$, specifies that the increase in ME's fatigue level as a result of the walking action will be half the distance it walks, where the user-defined function ($elapsed\_time$?) returns the time ME has spent on the current walking action assuming ME walks 1 distance unit per time step in the simulated world.

```
(setq answer_user_ynq
    (make-op :name 'answer_user_ynq :pars '(?q)
    :preconds '( (wants USER (that (tells ME USER (whether ?q)))) )
    :effects '( (not (wants USER (that (tells ME USER (whether ?q)))))
              (knows USER (that (answer_to_ynq? ?q)))
             )
    :time-required 1
    :value 10
    )
)
```

Figure 3.3: The $answer\_user\_ynq$ Model Operator

Not only does procedural attachment handle quantitative preconditions and effects (and the value and duration of an action, if applicable), but it also straightforwardly handles side-effects such as ME producing a printed answer to a question. For instance, consider the operator for answering a yes-no-question by the user in Figure 3.3. In the process of instantiating the effect ($knows$ $USER$ ($that$ ($answer\_to\_ynq$? ?q))) of this operator in ME's knowledge base, the evaluable term ($answer\_to\_ynq$? ?q) is evaluated by applying the user-defined LISP function $answer\_to\_ynq$? to the value bound to variable ?q. The function consults ME's knowledge base to determine an appropriate answer, such as (not (can_fly $guru$)) for the question (can_fly $guru$), and also outputs this answer to the user. Note that the particular effect stored for this example would be ($knows$ $USER$ ($that$ (not (can_fly $guru$)))), which correctly anticipates the user's resultant state of mind (assuming that the user "trusts" ME's answers).

Chapter 4

Reasoning about World States and Mental States

ME does not in general know all the current facts. Apart from geographical facts, it does not know the facts at other, unvisited locations, and even facts at visited locations may be occluded, or may have changed. Among the occluded facts we would normally have possessions and propositional attitudes of other entities (whereas ME presumably knows all of its own possessions and attitudes). Many traditional planning systems make a closed-world assumption (CWA), according to which absence (or non-inferrability) of a positive fact from the world model is viewed as supporting the negation of that fact. For example, if the world model does not contain ($has\ guru\ key1$), the negation (not ($has\ guru\ key1$)) is assumed. But this amounts to a tacit assumption that all positive facts (at least for the predicates of interest) are provided by ME's world model, contrary to the observations just made. Therefore, ME can only use a restricted version of the CWA; we do not want to abandon the CWA altogether, as it would be very inefficient to enumerate and maintain the very large number of negative predications that hold even in simple worlds.

Specifically, ME uses the CWA for every non-epistemic literal in which ME is the first ("subject") argument. (Epistemic literals, discussed below, are ones with predicate *knows*, any subject, and a that-clause or whether-clause as object.) For example, ME takes statements such as that ME is hungry, or has money, or likes Guru, or wants to answer the user's question, etc., to be false unless the corresponding positive literal appears in the current state. In other words, for such self-referential sentences ME's positive self-knowledge is assumed to be complete. But when the literal concerns a non-ME subject, ME requires explicit negative knowledge to draw a negative conclusion, except for the following two cases, where it applies the CWA:

1. literals about road connectivity and navigability (and even here the CWA could easily be weakened); e.g., the absence of ($road\ path5$) from ME's knowledge base, regardless of whether (not ($road\ path5$)) is present, would suffice for ME to conclude that $path5$ is not a $road$;

2. when (a) the subject is a local entity colocated with ME or one ME has visited, and (b) the predicate is non-occluded; e.g., a literal expressing that the piano is at ME's current location would be considered false unless explicitly present in ME's representation of the current state.

In all other cases concerning a non-ME subject, the mere absence of a literal from the world model is not interpreted as supporting its negation; in such cases, the negation may be explicitly known, or else its truth value may simply be unknown.

Algorithm 1 in Figure 4.1 formally summarizes the method of judging the truth value of a non-epistemic ground atom. A negative literal is judged to be true, false, or unknown accordingly as the atom that it negates is judged to be false, true, or unknown.

Epistemic literals require special treatment, because an agent that is committed to a proposition ψ should also be committed to the proposition that it *knows* that ψ, and that it *knows* whether ψ, without having to assert these conclusions separately. Similarly, if it is committed to the falsity of ψ, then it should also be committed to the falsity of the proposition that it knows that ψ, and to the truth of the proposition that it knows whether ψ. And finally, if it committed to neither the truth nor falsity of ψ, it should be committed to the propositions that it does not know whether ψ, does not know that ψ, and does not know that not ψ.

Thus ME judges the truth value of epistemic literals by an introspection algorithm rather than closed-world inference. The algorithm deals with both positive introspection (about knowledge ME possesses) and negative introspection (about knowledge it lacks). In particular, in the evaluation of a predication of form ($knows\ SUBJ$ ($that\ \psi$)) (e.g., ($knows\ ME$ ($that$ ($can\_talk\ guru$)))) with ψ being a literal, the algorithm considers the two cases $SUBJ = ME$ and $SUBJ \neq ME$. In the first case, ME recursively determines whether the literal ψ that is the object of knows-that is true, false, or unknown, and judges the autoepistemic predication to be true, false, or false respectively.

In the case $SUBJ \neq ME$, ME judges the epistemic predication as true if the predication is explicitly available in its world model, or $SUBJ$ is identical to the subject of ψ (thus

Algorithm 1 Evaluating Non-epistemic Atoms Using a Restricted CWA

Input: A non-epistemic ground atom *(P t_1 t_2...)*

Assumptions: t_1, t_2, ... are variable-free terms; each is either a constant, a reified proposition formed by applying *that* to a ground literal, a reified yes-no question formed by applying *whether* to a ground literal, or an *evaluable term* having been evaluated to a constant.

Output: *(P t_1 t_2...)*'s truth value (***true***, ***false***, or ***unknown***) in a given state (as modeled by *ME*)

```
if (P $t_1$ $t_2$...) is in the state description then return true

if (not (P $t_1$ $t_2$...)) is in the state description then return false

comment: To reach this point in the algorithm must mean that neither (P $t_1$ $t_2$...) nor (not (P $t_1$ $t_2$...))
         is in the state description.

if P is one of road, connects, navigable, ..., i.e., a geographic predicate to which the CWA applies
then return false

if $t_1$ = ME then return false

if P is unoccluded, and $t_1$ denotes an entity colocated with ME or at a place ME has visited
then return false

return unknown
```

Figure 4.1: Evaluating Non-epistemic Atoms Using a Restricted CWA

making an assumption that other agents are similar to itself in having complete positive self-knowledge – a type of simulative inference), and false otherwise. The latter assumption implies that other agents know only what ME *knows* they know.[1]

The method for predications of form ($knows\ SUBJ\ (whether\ \psi)$) is much the same. However, in the case of $SUBJ = ME$, when ψ is found to be true, false or unknown, the autoepistemic predication is judged to be true, true, or false respectively. In the case $SUBJ \neq ME$, ME again judges the epistemic predication as true only if $SUBJ$ is identical to the subject of ψ, and false otherwise. The method of evaluating epistemic predications is depicted in Algorithm 2 in Figure 4.2.

These inference capabilities are especially important in two respects. First, they are instrumental in ME's attempt to confirm or disconfirm action preconditions, including knowledge preconditions in actions like asking an agent whether ψ is true (viz., not knowing

[1]This negative closure assumption could be weakened, but we were more concerned with realistic epistemic modeling of ME than of other agents. For a formal treatment of simulative inference see [40].

```
Algorithm 2   Evaluating Epistemic Atoms Using Introspection

Input: An epistemic ground atom $\phi$ of the form (knows $t_1$ (that $\psi$)) or (knows $t_1$ (whether $\psi$))
Assumptions: $t_1$ is a constant, and $\psi$ is a ground (possibly epistemic) literal with any embedded
terms such as (+ account-balance? 100) having been evaluated to constants.
Output: $\phi$'s truth value (true, false, or unknown) in a given state (as modeled by ME)

if $\phi$ is in the state description then return true

if (not $\phi$) is in the state description then return false

comment: To reach this point must mean that neither $\phi$ nor (not $\phi$) is in the state description.

Evaluate $\psi$ to true, false, or unknown recursively using the method of Algorithm 1 or the method
of Algorithm 2, depending on whether $\psi$ is non-epistemic or not.

if $t_1$ = ME, or the subject of $\psi$ (the first argument appearing in the literal) is also $t_1$
  then
     if $\phi$ is of the form (knows $t_1$ (that $\psi$))
     then
          if value of $\psi$ is unknown then return false
          else return value of $\psi$
     else
          comment: $\phi$ is of the form (knows $t_1$ (whether $\psi$)).
          if value of $\psi$ is unknown then return false
          else return true

  else return false
```

Figure 4.2: Evaluating Epistemic Atoms Using Introspection

whether ψ is true, but believing that an agent knows whether ψ is true). Second, the inference capabilities are crucial in answering questions. Given a question, ME will either inform the user if it does not know the answer, or otherwise verbalize its answer(s) as English sentence(s). Extensive examples of ME's dialogue with the human user are in Section 5.2. They include knows-whether, knows-that, yes/no, and wh-questions in a simple simulated world set up in Section 5.1.

A more fine-grained type of restricted CWA, called the *local closed world (LCW)* was introduced by [34] and later adopted by [7]. This allows specification of the formulas that are subject to the CWA. While we could implement this approach, ours has the advantage of simplicity and was adequate for our experiments.

Besides its capacity for inferentially checking action preconditions and answering questions, ME also performs bounded forward inference for any state that it reaches in its simulated world or in its lookahead, based on all of its current factual knowledge and all of its general quantified knowledge. For example, from (*knows guru* (*that p*)), ME can infer both p and (*knows guru* (*whether p*)); from (*sasquatch moe*) and general knowledge ((*sasquatch* ?x) => (*has_IQ* ?x 50)) where variable ?x is assumed to be universally quantified over the domain, ME can infer that (*has_IQ moe* 50).

Chapter 5

An Experimental Agent

We implemented a version of ME in our self-motivated cognitive agent framework to demonstrate ME's question-answering as well as the benefits of ME's self-awareness and opportunistic behavior due to its deliberate, reward-seeking planning. We describe the simulated world, and how we implemented question-answering, in the ensuing two subsections.

5.1 Simulated World

ME is situated in a simulated world that is simple yet sufficiently complex to illustrate the distinctive features we mentioned. The simulated world consists of five named locations, connecting roads, and both animate entities (agents) and inanimate entities (objects) at various locations. Figure 5.1 depicts the simulated world. Specifically, road $path1$ connects locations $home$, $school$, and gym; road $path2$, locations $home$, $plaza$, and $school$; road $path3$, locations $home$, $company$, and $school$. Additionally, each road segment is annotated with its length in units. At the outset, object $pasta\_ingredients$, of cost 2.0, object $pepperoni\_pizza$, edible and of cost 6.0 by virtue of being of type $pizza$, and $apple\_juice$, potable and of cost 2.0 by virtue of being of type $juice$, are all at $plaza$; agents ME and $guru$, both communicative by virtue of being agents, are at $home$ and $school$, respectively; object $piano$, playable by virtue of being of type $music\_instrument$, is at $home$; object $self\_note$, readable by virtue of being of type $note$, is at $company$. Only predicates $contains$, $knows$, is_edible, and $is\_potable$ are designated as being occluded.

Initially, ME is penniless, has no swimming or cooking knowledge, is not tired, and has a hunger level of 4.0 and a thirst level of 2.0. In addition to knowledge of the road network and the existence of the object $piano$ at $home$ and its playability, ME knows whether $piano$

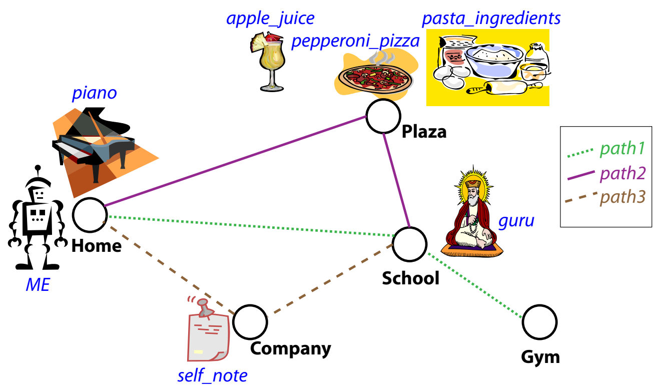


Figure 5.1: The Experimental World

is playable, the cost and location of *pasta_ingredients*, that *self_note* is readable and at *company*, and nothing more. *self_note* contains three pieces of knowledge, namely, that *guru* can talk, knows whether *apple_juice* is potable and whether *pepperoni_pizza* is edible.

The kinds of behavior we wish to enable in this simulated world include indulging in various pleasurable activities such as eating, drinking, cooking, swimming, and playing the piano. But actions have preconditions; e.g., eating and drinking require food to be at hand and knowledge about edibility and potability of certain items, cooking requires cooking knowledge as well as suitable ingredients and being at home, and swimming requires swimming knowledge and being at the gym. In turn, finding out about edibility and potability may require reading and/or consultation of the guru; obtaining ingredients for cooking may require buying them; and acquiring cooking and swimming skills by taking lessons may require working for money; we therefore enable all these activities, too.

To ensure that ME will have a general disposition towards action, we don't limit its reward system to bodily indulgence. Rather, ME reaps positive rewards from all of its activities, including answering user questions, working for money and taking cooking and swimming lessons. The only contributors to negative utility are hunger, thirst, fatigue, boredom and financial loss, but of course these are inevitable consequences of the passage of time or (in the last case) of purchases. Since in the stated initial conditions ME is penniless and can neither cook nor swim, we should expect to see behavior where ME works for money repeatedly, uses the earned money to take swimming and cooking lessons and buy ingredients, and eventually cooks meals, and of course partakes repeatedly of other rewarding activities – eating, drinking, sleeping, swimming, playing the piano, and answering user questions.

ME has the following operators (where the anticipated rewards are indicated parenthetically) at its disposal: *walk* (5 - ME's fatigue level), *eat* (4 * ME's hunger level), *drink* (4 * ME's thirst level), *work_and_earn_money* (10), *buy* (7 - the cost of the purchase), *cook* (7 * ME's hunger level), *swim* (12), *read* (7 per piece of knowledge contained that ME does not know), *play* (3), *answer_user_ynq* (10), *answer_user_whq* (10), *ask+whether* (5), *take_swimming_lesson* (10 - ME's fatigue level), *take_cooking_lesson* (10 - ME's fatigue level), and *sleep* (4 * ME's fatigue level).

In addition to the rewards and penalties associated with actions, there are also rewards for getting from a less desirable state to a more desirable one (e.g., where ME has a new possession), and penalties for getting from a more desirable state to a less desirable one (e.g., where ME is more fatigued). Generally only a small number of utility points are involved, but reaching a state where the user knows the answer to a question earns 50 points, while continuing to leave a question unanswered costs 20 points.

Any potentially multi-step action is subject to possible interference by two types of exogenous events, namely fire and rain, although only fire may disrupt the action (e.g., traveling). A spontaneous fire has a 5% chance per time step of starting; once started, it has a 50% chance per time step of dying by itself, and it is extinguished as soon as there is rain. Spontaneous rain has a 33% chance of beginning; once it has begun, it has a 25% chance of stopping.

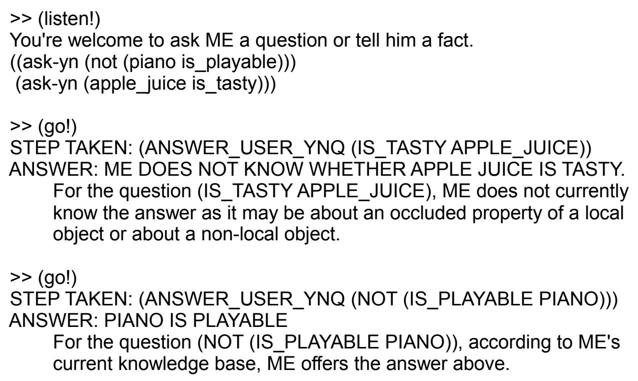
5.2 Question-Answering

ME's question-answering behavior is handled uniformly via ME's planning and procedural capabilities. Accordingly, the projective lookahead identifies a seemingly best action for ME to take at each step; thus, to answer a user question, ME must choose to do so as a/the seemingly most rewarding action. In support of our vision of ME as a dialogue agent, we induce its curiosity and helpfulness by making ME favor acquiring knowledge and answering questions as rewarding actions.

Using interface command (*listen*!), the user can enable transmission of yes/no or wh-questions or knowledge tidbits to ME. While ME can't help "hearing" a question posted to it via (*listen*!), its response, if any, is dependent on its deliberations. The effect on ME when the user subsequently supplies a fact or question is just the appearance of the new fact, or the user's desire for an answer to the question, in its KB; its inferences and planning are affected accordingly.

Knowledge tidbits are supplied as ground predications or negated ground predications, and are rejected by the system (with an appropriate message) if inconsistent with the "actual" world. Supplying a fact already known to ME has no effect (except to produce a message).

Given a yes/no question of the form *(ask-yn q)*, ME must consider two cases. If ME knows either q or $\neg q$ to be true, then ME will impart that knowledge as an English sentence. On the other hand, if ME knows neither q nor $\neg q$ to be true, then it must be that q is about a nonlocal entity or about an occluded property of a local non-ME entity, and ME informs the user accordingly. The example in Figure 5.2 illustrates ME's responses for the two cases of yes/no questions. (*go*!) is a command that activates ME for one planning and action cycle, i.e., ME wil consider its applicable actions in the current state and evaluate contemplated action sequences, select a/the seemingly best action to perform, and then perform the chosen action to achieve the effects and derive further inferences.

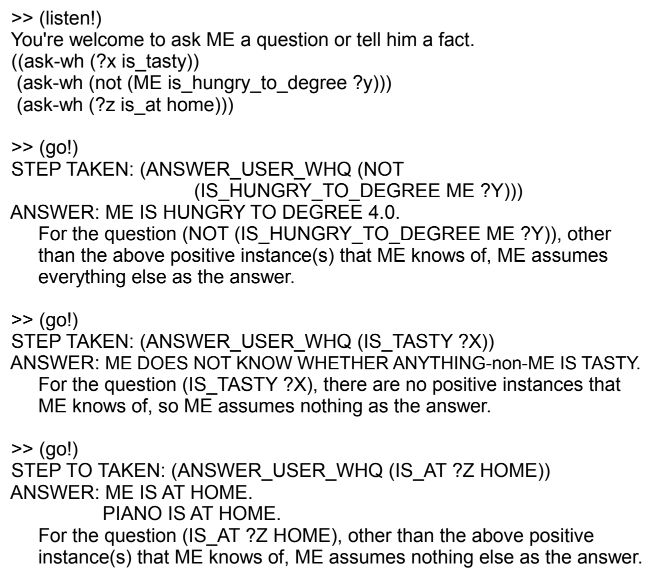


```
>> (listen!)
You're welcome to ask ME a question or tell him a fact.
((ask-yn (not (piano is_playable)))
 (ask-yn (apple_juice is_tasty)))

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ (IS_TASTY APPLE_JUICE))
ANSWER: ME DOES NOT KNOW WHETHER APPLE JUICE IS TASTY.
      For the question (IS_TASTY APPLE_JUICE), ME does not currently
      know the answer as it may be about an occluded property of a local
      object or about a non-local object.

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ (NOT (IS_PLAYABLE PIANO)))
ANSWER: PIANO IS PLAYABLE
      For the question (NOT (IS_PLAYABLE PIANO)), according to ME's
      current knowledge base, ME offers the answer above.
```

Figure 5.2: Answering *yes/no* Questions

A wh-question of the form *(ask-wh r)* must have at least one variable (which is indicated by prefix ?) in r. For instance, (*is_at* ?x ?y) translates to "where is every entity located?" while (*not* (?x *is_bored*)) corresponds to the question "who is not bored?". In computing the answer(s) to r, ME attempts to unify r with facts in ME's current knowledge base; for each set s of bindings found for the variables in r, ME formulates the corresponding answer by replacing the variables in r with bindings in s. As Figure 5.3 illustrates, ME uses the restricted CWA (see Chapter 4) in verbalizing its responses.

Figure 5.4, including knows-whether and knows-that questions, is a concatenation of



```
>> (listen!)
You're welcome to ask ME a question or tell him a fact.
((ask-wh (?x is_tasty))
 (ask-wh (not (ME is_hungry_to_degree ?y)))
 (ask-wh (?z is_at home)))

>> (go!)
STEP TAKEN: (ANSWER_USER_WHQ (NOT
                              (IS_HUNGRY_TO_DEGREE ME ?Y)))
ANSWER: ME IS HUNGRY TO DEGREE 4.0.
    For the question (NOT (IS_HUNGRY_TO_DEGREE ME ?Y)), other
    than the above positive instance(s) that ME knows of, ME assumes
    everything else as the answer.

>> (go!)
STEP TAKEN: (ANSWER_USER_WHQ (IS_TASTY ?X))
ANSWER: ME DOES NOT KNOW WHETHER ANYTHING-non-ME IS TASTY.
    For the question (IS_TASTY ?X), there are no positive instances that
    ME knows of, so ME assumes nothing as the answer.

>> (go!)
STEP TO TAKEN: (ANSWER_USER_WHQ (IS_AT ?Z HOME))
ANSWER: ME IS AT HOME.
             PIANO IS AT HOME.
    For the question (IS_AT ?Z HOME), other than the above positive
    instance(s) that ME knows of, ME assumes nothing else as the answer.
```

Figure 5.3: Answering *wh* Questions

several dialogue exchanges between ME and the user, showing only ME's actions to answer user questions and ME's corresponding verbalized English answers. For clarity, we further annotate some action-answer pairs with justifications for the answers. These examples demonstrate ME's ability to introspect positively and negatively using the restricted CWA (see Chapter 4).

```
>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ
             (KNOWS GURU (WHETHER (LIKES ME PASTA))))
ANSWER: ME DOES NOT KNOW WHETHER GURU KNOWS WHETHER ME LIKES PASTA.
//Reason: Guru's knowledge is still occluded from ME since ME has not acquired
//such information about Guru.

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ (GURU KNOWS TWEETY))))
ANSWER: ME DOES NOT KNOW WHETHER GURU KNOWS TWEETY.
//Reason: Guru's knowledge is still occluded from ME since ME has not acquired
//such information about Guru.

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ
             (KNOWS GURU (WHETHER (KNOWS GURU TWEETY))))
ANSWER: GURU KNOWS WHETHER GURU KNOWS TWEETY.
//Reason: Although ME does not know whether Guru likes Tweety, Guru must know
//this about itself by simulative inference.

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ (CAN_DANCE GURU))
ANSWER: IT IS NOT THE CASE THAT GURU CAN DANCE.
//Reason: ME knows the answer since ME has visited Guru and Guru's dancing
//ability is not an occluded property.

>> (go!)
STEP TAKEN: (ANSWER_USER_WHQ (IS_AT ?X ?Y))
ANSWER: ME IS AT GYM.
        GURU IS AT SCHOOL.
        SELF NOTE IS AT COMPANY.
        PASTA INGREDIENTS IS AT PLAZA.
        PIANO IS AT HOME.
For the question (IS_AT ?X ?Y), other than the above positive instance(s) that ME
knows of, ME assumes nothing else as the answer.

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ (CAN_DANCE ME))
ANSWER: IT IS NOT THE CASE THAT ME CAN DANCE.

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ (NOT (IS_NAVIGABLE PATH3)))
ANSWER: PATH3 IS NAVIGABLE.

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ
             (KNOWS GURU (THAT (IS_NAVIGABLE PATH3))))
ANSWER: ME DOES NOT KNOW WHETHER GURU KNOWS THAT PATH3 IS NAVIGABLE.
//Reason: Even though ME indeed knows that path3 is navigable, since ME has not
//learned of Guru's knowledge about the navigability of path3, this particular
//piece of Guru's knowledge is still occluded from ME.

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ (NOT (IS_BORED ME)))
ANSWER: IT IS NOT THE CASE THAT ME IS BORED.

>> (go!)
STEP TAKEN: (ANSWER_USER_YNQ (KNOWS GURU (THAT (IS_BORED ME))))
ANSWER: IT IS NOT THE CASE THAT GURU KNOWS THAT ME IS BORED.
```

Figure 5.4: Answering Questions Using Introspection

Chapter 6

Results

We present empirical results of three distinct cases illustrating the advantages of ME's self-awareness and opportunistic behavior, resulting from its self-motivated, deliberate, continual planning, in the context of the simulated world described in Section 5.1. The first case is ME's default, self-aware and opportunistic behavior; the second case, its unthinking and non-self-aware behavior by ablating its projective lookahead; the third case, its purely goal-directed behavior by ablating its opportunistic tendencies.

With the exception of the purely goal-directed case, each case includes ten independent runs of 40 *steps* apiece, where at each *step* ME chooses and executes a/the next best action. The lookahead used in each case is three levels (i.e., three steps into the future from the current state) and has branching factors of 5, 4, 3 at the three successive levels. Variable behaviors could result from random occurrences of fire and rain, or from randomly and with uniform probability selecting a best action (among equally attractive best action candidates) to perform at a step. Therefore, we show the average in comparing these three different cases.

6.1 Opportunistic Behavior

We report experiments in which ME exhibits (default) opportunistic and foresighted behavior with all its faculties intact. The success of a run is measured in terms of the net utility (NU), accumulated over the entire sequence of actions and corresponding states attained.

For the opportunistic case, we conducted ten independent runs of ME's opportunistic behavior, using a three-level lookahead with branching factors of 5, 4, and 3 for the three successive levels. Each run was for 40 steps. ME achieved an NU of 1260.85 averaged over the ten runs ranging from 1204.5 to 1389.0.

As an example, Figure 6.1 shows ME's sequence of actions for one run, where ME's NU was 1288.0. Each action, which is the inner parenthesized expression, was logged with a number indicating which iteration of this action was listed, its anticipated duration, and the time in the simulated world when that iteration began. An inner parenthesized expression represents an action performed by ME, and it consists of the action name as well as its specific input arguments. To save space required for enumerating all iterations, for a multi-step action (i.e., one that ran for more than one iteration), we only enumerate its first iteration and its final iteration on the same line; i.e., each multi-step action below, such as ($WORK\_AND\_EARN\_MONEY$ 4.0 0.0 1.0) on the 2nd line running for five iterations, has exactly one pair of listings on the 2nd line. On the other hand, one-step action below, such as ($READ$ 6.5 $SELF\_NOTE$ $COMPANY$) on the 3rd line, has exactly one listing.

```
((WALK HOME COMPANY PATH3 0.0) 1 2 0)          ((WALK HOME COMPANY PATH3 0.0) 2 2 1)
((WORK_AND_EARN_MONEY 4.0 0.0 1.0) 1 5 3)      ((WORK_AND_EARN_MONEY 4.0 0.0 1.0) 5 5 7)
((READ 6.5 SELF_NOTE COMPANY) 1 1 9)
((WALK COMPANY SCHOOL PATH3 6.0) 1 3 11)       ((WALK COMPANY SCHOOL PATH3 6.0) 3 3 13)
((ASK+WHETHER GURU ((IS_POTABLE APPLE_JUICE) SCHOOL) 1 1 15)
((TAKE_COOKING_LESSON 0.0 7.5) 1 4 17)         ((TAKE_COOKING_LESSON 0.0 7.5) 4 4 20)
((TAKE_COOKING_LESSON 4.0 9.5) 1 4 22)         ((TAKE_COOKING_LESSON 4.0 9.5) 4 4 25)
((TAKE_COOKING_LESSON 8.0 11.5) 1 4 27)        ((TAKE_COOKING_LESSON 8.0 11.5) 4 4 30)
((TAKE_COOKING_LESSON 12.0 13.5) 1 4 32)       ((TAKE_COOKING_LESSON 12.0 13.5) 4 4 35)
((TAKE_COOKING_LESSON 16.0 15.5) 1 4 37)       ((TAKE_COOKING_LESSON 16.0 15.5) 4 4 40)
((ASK+WHETHER GURU ((IS_EDIBLE PEPPERONI_PIZZA) SCHOOL) 1 1 42)
((WALK SCHOOL HOME PATH1 17.5) 2 2 45)         ((WALK SCHOOL HOME PATH1 17.5) 1 2 44)
((PLAY 2.0 18.5 PIANO HOME) 1 1 47)
((SLEEP 19.0 7.0) 1 38.0 49)                   ((SLEEP 19.0 7.0) 38 38.0 86)
((WALK HOME SCHOOL PATH1 0.0) 1 2 88)          ((WALK HOME SCHOOL PATH1 0.0) 2 2 89)
((WALK SCHOOL GYM PATH1 1.0) 1 1 91)
((TAKE_SWIMMING_LESSON 16.5 0.0 1.5 2.5) 1 3 93)     ((TAKE_SWIMMING_LESSON 16.5 0.0 1.5 2.5) 3 3 95)
((TAKE_SWIMMING_LESSON 18.0 6.0 4.5 3.5) 1 3 97)     ((TAKE_SWIMMING_LESSON 18.0 6.0 4.5 3.5) 3 3 99)
((TAKE_SWIMMING_LESSON 19.5 12.0 7.5 5.0) 1 3 101)  ((TAKE_SWIMMING_LESSON 19.5 12.0 7.5 5.0) 3 3 103)
((WALK GYM SCHOOL PATH1 10.5) 1 1 105)
((WALK SCHOOL GYM PATH1 11.0) 1 1 107)
((WALK GYM SCHOOL PATH1 11.5) 1 1 109)
((WALK SCHOOL HOME PATH1 12.0) 1 2 111)           ((WALK SCHOOL HOME PATH1 12.0) 2 2 112)
((WALK HOME PLAZA PATH2 13.0) 1 2 114)            ((WALK HOME PLAZA PATH2 13.0) 2 2 115)
((BUY 15.0 PASTA_INGREDIENTS PLAZA 2.0) 1 1 117)
((WALK PLAZA HOME PATH2 14.0) 1 2 119)            ((WALK PLAZA HOME PATH2 14.0) 2 2 120)
((COOK 21.0 20.0 15.0)  1 1 122)
((EAT 21.0 PASTA) 1 1 124)
((SLEEP 16.0 0.0) 1 32.0 126)                     ((SLEEP 16.0 0.0) 32 32.0 157)
((WALK HOME PLAZA PATH2 0.0) 1 2 159)             ((WALK HOME PLAZA PATH2 0.0) 2 2 160)
((BUY 13.0 PASTA_INGREDIENTS PLAZA 2.0) 1 1 162)
((WALK PLAZA HOME PATH2 1.0) 1 2 164)             ((WALK PLAZA HOME PATH2 1.0) 2 2 165)
((COOK 8.0 20.0 2.0) 1 1 167)
((EAT 8.0 PASTA) 1 1 169)
((WALK HOME PLAZA PATH2 3.0) 1 2 171)             ((WALK HOME PLAZA PATH2 3.0) 2 2 172)
((BUY 11.0 APPLE_JUICE PLAZA 2.0) 1 1 174)
((DRINK 6.5 APPLE_JUICE) 1 1 176)
((WALK PLAZA HOME PATH2 4.0) 1 2 178)             ((WALK PLAZA HOME PATH2 4.0) 2 2 179)
((WALK HOME COMPANY PATH3 5.0) 1 2 181)           ((WALK HOME COMPANY PATH3 5.0) 2 2 182)
((WORK_AND_EARN_MONEY 0.0 9.0 6.0) 1 5 184)      ((WORK_AND_EARN_MONEY 0.0 9.0 6.0) 5 5 188)
```

Figure 6.1: Example of an Opportunistic Run

We reiterate that ME's chosen seemingly best action need not be considered immediately rewarding by ME; however, it is chosen because it constitutes the first action in the/a most rewarding sequence of actions (plan) that ME can foresee as executable and attainable from ME's current state. For instance, although ME might not receive a reward for walking from *home* to *plaza*, but ME may very well foresee a substantial reward resulting from traveling to *plaza* to buy *pasta_ingredients*, taking *pasta_ingredients* back *home*, cooking with *pasta_ingredients* to produce *pasta*, and ultimately eating the cooked *pasta*. Therefore, the use of a reasoned projective lookahead enables ME to exhibit both foresight and opportunism in its behavior.

The relatively high NU here compared with the NU for single-minded goal pursuit (below) results from ME's seizing various favorable opportunities encountered while exploring the simulated world. The opportunities seized include working to earn money, sleeping to relieve fatigue, playing *piano* to assuage boredom (boredom is an assumed effect of reading, though ME is not initially bored), taking cooking lessons to learn to cook, and several others.

6.2 Unthinking, Non-Self-Aware Behavior

To simulate unthinking, non-self-aware behavior, we suppressed ME's projective lookahead and instead made ME randomly choose applicable actions (with uniform probability) in any given state, without consideration of consequences or anticipated utility. We can interpret the resulting behavior as showing the consequences of abandoning self-awareness. After all, much of ME's self-awareness is comprised of ME's knowledge about its own situation and about its action preconditions and effects, and its anticipation of the situations it may get into when performing a sequence of actions, and of the rewards and penalties associated with those situations and actions.

The average NU over these ten runs of 40 steps apiece is -627.65, and ME's behavior was extremely haphazard, yielding greatly varied NUs ranging from -1465.0 to -56.0. The average NU here, though unreliable, is significantly lower than in the opportunistic, reasoned, self-aware case in Section 6.1. Perhaps more importantly, the very low NU values obtained in some cases show that the unthinking behavior is extremely risky. These results are unlikely to surprise the reader, but they are not a foregone conclusion: one can certainly imagine environments in which random behavior is as good as any other. But it is precisely in worlds where certain formally statable regularities constrain the situations encountered by an agent and the effects produced by its actions, that thinking and self-awareness are beneficial.

6.3 Goal-Directed Behavior

For a self-motivated, reward-seeking agent to single-mindedly pursue a particular goal, when it *could* exploit various other intrinsically rewarding actions and states, it must in fact be "deluded": It must operate as if the goal were much more rewarding than the pleasurable actions and states it is foregoing. But the only way to implement such a delusional agent and observe its actions, while not changing its basic reward-seeking architecture, is to change the rewards that the ME believes it will reap.

Thus to simulate purely goal-directed behavior, we skewed MEs "subjective" rewards and penalties for actions and states away from their true values in the lookahead calculation (while evaluating actual rewards just as in the opportunistic case.) Specifically, in order to make eating self-prepared *pasta* ME's sole goal, we used the following distorted settings, favoring courses of action leading to the goal: (a) The anticipated rewards of *eat*, *take_cooking_lesson*, *buy*, *cook*, and *work_and_earn_money* are each set to 50, while the anticipated rewards of all others are nullified; and (b) the acquisition of cooking knowledge, money, *pasta_ingredients*, and *pasta* in a state reached are rewarded, and so is the consumption of *pasta* or *pasta_ingredients* in a state reached, and so is any decrease in hunger in a state reached, while an increase in hunger in a state reached is penalized, and all other changes in a state reached are ignored.

We had to assign positive values to actions and states that are needed to reach the goal of eating pasta, rather than simply rewarding the latter. The reason is that nontrivial problem-solving in combinatorially complex domains is impractical without heuristics. Most practical planners use heuristics that either estimate the distance to the goal, or (inversely) measure the similarity of a given state to the goal state. The method we have employed here is particularly simple, since it merely biases the agent towards types of actions likely to lead to the goal. This is sufficient to solve the pasta-eating problem, even though the solution is 14-25 steps long (e.g., involving multiple cooking lessons).

Given the goal of eating self-cooked *pasta*, ME's sequence of at least 14 actions was generally found with few missteps. The NU was either 197 or 187 in all cases (because of a random route choice), and 193 on average. Figure 6.2 shows a run that yielded an actual NU of 187.0.

The rationale behind ME's sequence of actions to reach the goal is easily understood by considering them anti-chronologically. To eat ME-cooked *pasta*, ME must cook to produce *pasta*. Since cooking *pasta* required the use of *pasta_ingredients*, which ME did not initially possess, ME must travel to *plaza*, the initial location of *pasta_ingredients*, and

```
((WALK HOME SCHOOL PATH1 0.0) 1 2 0)              ((WALK HOME SCHOOL PATH1 0.0) 2 2 1)
((TAKE_COOKING_LESSON 0.0 1.0) 1 4 3)             ((TAKE_COOKING_LESSON 0.0 1.0) 4 4 6)
((TAKE_COOKING_LESSON 4.0 3.0) 1 4 8)             ((TAKE_COOKING_LESSON 4.0 3.0) 4 4 11)
((TAKE_COOKING_LESSON 8.0 5.0) 1 4 13)            ((TAKE_COOKING_LESSON 8.0 5.0) 4 4 16)
((TAKE_COOKING_LESSON 12.0 7.0) 1 4 18)           ((TAKE_COOKING_LESSON 12.0 7.0) 4 4 21)
((TAKE_COOKING_LESSON 16.0 9.0) 1 4 23)           ((TAKE_COOKING_LESSON 16.0 9.0) 4 4 26)
((WALK SCHOOL COMPANY PATH3 11.0) 1 3 28)         ((WALK SCHOOL COMPANY PATH3 11.0) 3 3 30)
((WORK_AND_EARN_MONEY 4.0 0.0 12.5) 1 5 32)       ((WORK_AND_EARN_MONEY 4.0 0.0 12.5) 5 5 36)
((WALK COMPANY SCHOOL PATH3 17.5) 1 3 38)         ((WALK COMPANY SCHOOL PATH3 17.5) 3 3 40)
((WALK SCHOOL PLAZA PATH2 19.0) 1 4 42)           ((WALK SCHOOL PLAZA PATH2 19.0)  4 4 45)
((BUY 15.0 PASTA_INGREDIENTS PLAZA 2.0) 1 1 47)
((WALK PLAZA HOME PATH2 21.0) 1 2 49)             ((WALK PLAZA HOME PATH2 21.0) 2 2 50)
((COOK 6.5 20.0 22.0)  1 1 52)
((EAT 6.5 PASTA)  1 1 54)
```

Figure 6.2: Example of a Goal-Directed Run

buy *pasta_ingredients* there. Similarly we find that ME must travel to *company* and work to earn money, and initially must travel to school to take a sufficient number of cooking lessons.

The average NU in this case was 193.0, with considerable disutility coming from ME's growth in hunger and fatigue (ignored in its "delusional" state). This is considerably lower than the average of 1260.85 in ME's opportunistic behavior in Section 6.1. As enumerated at the end of that section, instead of doggedly pursuing the sole goal of eating *pasta*, ME was able to correctly evaluate and exploit *additional* opportunities, such as sleeping to relieve fatigue, playing *piano* to assuage boredom, taking swimming lessons to learn to swim, gaining knowledge from reading and from *guru*, and eating and drinking foods other than *pasta*. The results establish that ME benefits from awareness and exploitation of opportunities for immediate rewards, and not just distant ones.

Chapter 7

Comparisons with Some Classical, Contingent, and Continual Planners

While we emphasize that our framework is intended as a platform supporting certain types of cognitive agents that use planning to accumulate utility, rather than as a platform supporting state-of-the-art goal-directed planners, it is nonetheless of interest to make some comparisons with classical planners and more recent conformant, contingent, and continual planners.

7.1 Classical Planning

Classical planners (e.g., [82]) are purely goal-directed, aiming all their actions at attainment of some future goal state. While they may attempt to minimize the cost of reaching a goal state, they have no notion of intrinsically rewarding actions or situations that could be exploited along the way to the goal. They are also apt to assume that the planner's knowledge of the search space to be explored is sufficiently complete in all the relevant respects to allow advance "mental" construction of a complete plan that provably leads to the goal state.

This is rather different from planning in everyday life (including dialogue interactions), where unanticipated events and situations, presenting new opportunities for rewards or new risks or costs, can arise regularly. That is why allowing for opportunistic responses to situations encountered in the "world" was an essential desideratum in the design of our framework.

However, ascertaining that classical planning problems can be tackled within our framework would provide evidence for the versatility of our reward-based framework. We thus undertook experiments using the classical Logistics domain and the 3-disk and 4-disk Towers of Hanoi problems. In these problems, inept moves can easily undo gains made previously, making them combinatorially more complex than the example of goal-directed behavior (aimed at pasta-eating) we previously illustrated in our more stable Gridworld. (Note that in the pasta-eating task, the various preparatory steps have persistent effects: once ME has gained some cooking knowledge, this knowledge is not lost; once ME has pasta ingredients, this remains true till the ingredients are used to make pasta; and so on.) The encoding of the problems is based on a fixed locale, as if the agent were performing paper-and-pencil problem solving; also the full CWA holds. The issue requiring some thought is how to incorporate search heuristics into our agent's view of the world, which is reward-based, not directly goal-driven.

In the Logistics world, there are two cities, each containing an airport and a post office. At least one truck serves each city, and an airplane, located at either of the airports, can commute between the cities. The goal is to deliver one or more parcels to designated locations. We experimented with problems requiring from 3 to 13 steps. Our approach was to use a uniform A*-like heuristic, (under)estimating the remaining number of steps. An effective way of incorporating the heuristic function into the agent's reward system is to make dual use of it: In any state, the agent receives a negative reward proportional to its estimate of remaining distance to the goal state, and for any action it receives a negative reward if the action fails to reduce that estimated distance. Seemingly helpful actions carry 0 reward, and we also allow a 0-reward *do-nothing* action that prevents pointless vehicle movements once the goal is achieved. With this setup and compiled Lisp, our agent solved each of five Logistics problems requiring 3, 6, 9, 10, and 13 steps in under 0.4 seconds on a standard desktop computer, employing a 2-step search horizon.

We implemented the 3-disk and the 4-disk versions of the Towers of Hanoi puzzle. A weak yet sufficient heuristic we adopted looked solely at the positions of disks on peg 3 in any given state, rewarding each disk that is in its final place in the disk stack on peg 3, and symmetrically punishing undoing a disk that is already in its final place in the stack of disks on peg 3. The rewards and symmetric penalties are proportional to the size of the disk. The only available actions were a 0-utility *move* action and a 1-utility *do-nothing* action, with the latter being applicable only when all disks have been correctly placed on peg 3.

For the 3-disk version, we instituted a 4-step search horizon with the best 3-action breadth at each depth, and our agent invariably reached the goal state in the optimal 7 steps, without missteps and taking only about 0.31 seconds (averaged over 20 runs). For the 4-disk version, using an 8-step deep search horizon with the best 3-action breadth at each depth, our agent always reached the goal state in the optimal 15 steps, without missteps and taking about 55.35 seconds (averaged over 20 runs). The average 55.35 seconds for 4-disk seemed rather slow, but understandably so, since the forward search tree at each step contained up to 9840 states, implying rampant duplication of the 81 distinct states in this puzzle.

We could have obtained faster solutions by avoiding state duplication in the forward search. But our goal was merely to demonstrate that our agent framework is flexible enough to allow representation and solution of classical planning problems, rather than to match the performance of state-of-the-art algorithms such as Graphplan, SATPLAN, and FF. We also note that the avoidance of state duplication is more of an issue in solving combinatorial puzzles than in guiding the behavior of a reward-seeking agent in a world where rewards are distributed over many states. For example, the optimal behavior for an agent similar to the one in our cooking world may well be to loop indefinitely through a sequence of actions and states where the means for earning rewards are repeatedly acquired, deployed, and depleted.

7.2 Continuous Planning: The Colorballs-n-x Problem

Our framework shares an affinity with conformant and contingent planners (e.g., [9, 11, 16]) in looking beyond the flawed presupposition made by most classical planners of possessing correct and complete world information. Conformant and contingent planners emerged to tackle planning in the presence of incomplete information, namely, uncertainty about the initial state or action effects. [1] extended the approach of [63] to translate a contingent planning problem P into a classical planning problem to then be solved by a classical planner (e.g., FF), although assuming that P involves uncertainty in the initial state only and that all actions are deterministic. The resulting Closed-Loop Greedy (CLG) planner introduced in [1] can be used either on-line to capture single executions, or off-line to construct full contingent plans.

Contingent planners typically rely on constructing full contingent plans that will assure mapping of an initial belief state into the target belief state, despite the incompleteness of the beliefs involved. However, the size of the solutions, exponential in the number of possible observations, is often a computational bottleneck. The data in the 2nd - 7th columns in Table 7.1 were extracted from [1] running on a Linux machine at 2.33 GHz with 2Gb of RAM with a cutoff of 45 minutes and 1.8Gb of memory. Each *time* column shows the total time taken in seconds by the corresponding planner on a particular Colorballs-n-x problem, and each *#acts* column displays the total number of actions in the solution. Notably, Contingent-FF [37] failed to solve 3 problems after spending 45 minutes on each, Pond 2.2 [16] failed to solve 2 problems after exhausting the available memory on each, and CLG also timed out on Colorballs-10-2 after 45 minutes.

We conducted our experiments on a similar Linux machine at 2.40GHz with 2Gb of RAM. The numerical data we show in Table 7.1's last two columns under the planner heading SCAF, standing for our self-motivated cognitive agent framework, are the averages over 50 randomly generated configurations matching the particular Colorballs-n-x settings. In the scenario world, the agent's goal in its n by n gridworld is to carry each of the x balls from its initial position to its goal position based on the ball's color in the gridworld, and each ball is of one of the only four possible colors red, blue, yellow, and green. The agent's repertoire of action operators is composed of:

- *walk* from a location to one of its adjacent locations, with an expected net utility increase equal to the agent's degree of happiness;
- *pick-up* a ball from a location other than the four color-based goal positions, with an expected net utility increase of 100;
- *put-down-red, put-down-blue, put-down-yellow, put-down-green*, i.e., put down a ball of the appropriate color, where this yields a 100-point utility gain if done at the target location for that color;
- *announce-success* when the agent has placed all balls in their designated color-based locations, with an expected net utility increase of 100.

Other than the anticipated utility values of actions operators, states present no anticipated values to the agent in the Colorballs gridworld. Our agent uses a 3-step deep search horizon with a branching factor of 4 at each step, but aside from this projective lookahead, our agent uses absolutely no heuristics or additional search strategies.

Table 7.1 shows results for five instances of Colorballs-n-x, each averaged over 50 trials. (All times in this and subsequent tables are in seconds.) The results compare very favorably with the other three contingent planners, which either timed out after 45 minutes or exceeded available memory on several problems.

Table 7.1: Working with or without Full Contingent Plans

Problem	**Contingent FF**		**Pond**		**CLG**		**SCAF**	
	time	#acts	time	#acts	time	#acts	time	#acts
cb-4-1	0.27	277	0.98	102	0.35	295	6.31	22.18
cb-4-2	35.88	18739	40.92	1897	18.83	20050	8.70	36.14
cb-4-3	T		1063.11	28008	1537.99	1136920	11.72	45.14
cb-10-1	T		M		415.73	4445	313.89	246.94
cb-10-2	T		M		T		696.27	484.64

Table 7.2: CLG in Execution Mode on Colorballs-9-i

Problem	**CLG in Execution Mode**					
	Translation		Search		#acts	
	time	size (MB)	avg	max	avg	max
cb-9-1	20.9	16.5	1.21	7.80	33.7	197
cb-9-2	56.4	33.7	4.84	25.70	57.1	288
cb-9-3	113.7	51.4	46.26	122.19	76.3	367

Table 7.3: Our Framework on Colorballs-9-i

Problem	**SCAF**			
	Run Time		#acts	
	avg	min / max	avg	min / max
cb-9-1	150.30	4.57 / 516.61	168.5	5 / 543
cb-9-2	281.38	16.37 / 642.90	239.12	11 / 552
cb-9-3	345.33	62.60 / 799.17	333.58	51 / 694

Table 7.2 shows the noteworthy performance of CLG on a set of Colorballs-9-*i* problems, in online mode (as opposed to contingent plan construction mode). Table 7.3 shows our empirical data on the same problems; again averaging was done over 50 randomized runs per Colorballs instance. Our runs take several times longer (by a factor that diminishes as the number of balls increases). Our advantages are that no translation is required for the scenario world, and that the file size of our scenario world for each Colorballs-9-*i* problem is under 16 KB (including coding comments) – as *i* increases by 1, an additional (*place-object* ...) call suffices. However, our higher run times and number of actions indubitably included meandering actions that repeatedly visited the same states. Again, as we noted earlier in our Towers of Hanoi discussion, this kind of behavior is often beneficial for a reward-seeking agent, the kind exemplified by our framework. But the above results again demonstrate the versatility of our framework, showing that it even enables solution of problems that thwart state-of-the-art contingent planners that must build full contingent plans.

7.3 Continuous Multiagent Planning: The Multiagent-n-x-b Traveling Domain

Recent years have seen growing interest in continual planning, which interleaves planning, execution and monitoring in a dynamic, only partially observable, multiagent environment. [15] introduced a simulation environment for Continual Multiagent Planning, called MAPSIM, in which an agent can directly execute planned actions in a simulation, acquire perceptions specified by the planning domain ontology, and use these perceptions for plan monitoring and replanning. The authors implemented each agent as an individual and independent MAPSIM process, and thus, each agent knows only about its own goal state and its own perceptions. They also devised a Multiagent-n-x-b domain, similar to the *grid* domain in the International Planning Competitions. In the Multiagent-n-x-b domain, there is an n by n grid with b blocked locations, which cannot be traversed by any agent, and there are x agents each of which is to travel from its initial location to its goal location in the gridworld.

To assess the capabilities of our framework in this domain, we made some extensions to allow for coexisting self-motivated agents, where each has its own set of parameters, knowledge base, model of the world, etc. – i.e., all the components and capabilities of a separate gridworld agent, but sharing the "actual" world with the other agents.

In the Multiagent-n-x-b domain, each agent starts out unaware of the existence of any other agents (let alone their locations), unaware of any blocked locations; but knowledgeable only about the adjacency information of locations in the gridworld (i.e., which are the 4 neighbor locations for each location), about its initial location and co-located local entities, and what its goal location is. All agents take turns identifying and executing a seemingly best action for their own utility optimization. Since at most one agent is allowed to occupy a location, immediately prior to identifying and executing a seemingly best action, the agent will do a look-around solely to learn whether each adjacent location is occupied (i.e., either one of the b blocked locations, or occupied by at least one entity). Despite this look-around, the agent still may not know of all the entities at an adjacent location until it actually reaches it. Furthermore, the look-around gives the *is_occupied*? information of its adjacent locations only, not that of all the locations in the gridworld, so the agent may still retain absolutely no or outdated *is_occupied*? information of all other locations.

Each agent can stay put at any step, but this is dispreferred to walking unless the agent has reached its goal location, at which point staying put earns a large utility increase. Each agent's repertoire of action operators is composed of:

- *walk* from a location to one of its unoccupied adjacent locations, with an expected net utility increase either of 10 if the new location is the agent's goal location, or of 0 otherwise;

- *stay-put*, with an expected net utility increase of 10 if the current location is the agent's goal location, or of -1 otherwise.

Other than the anticipated utility values of actions operators, states present no anticipated values to the multiple agents in this domain. Each agent uses a 4-step deep search horizon with branching factors of 4, 4, 3, and 2 at successive steps; aside from this projective lookahead, each agent uses no heuristics or additional search strategies.

Table 7.4shows our results for the Multiagent-n-b-x domain, for 22 different setting of the n-b-x parameters, again with averaging over 50 randomized instances each (avoiding unsolvable configurations).

Once an agent reaches its goal location, it much prefers staying put to walking about. As in [15], there is no inter-agent communication, coordination or collaboration, so it is possible for an agent to become trapped at a non-goal location when all its adjacent locations are occupied (i.e., either being a blocked location, or occupied by another agent similarly trapped itself or already in its goal location). Accordingly, using our knowledge of the size of the configuration, we imposed a sufficiently large upper bound on the number of actions and cutoff time limit for each agent, in order to filter out theoretically workable

Table 7.4: Our Framework on 22 Multiagent-n-x-b Problems

Problem	**SCAF**			
	Run Time		#acts	
	avg	min / max	avg	min / max
ma-6-4-10	15.63	2.19 / 63.10	70.86	10 / 288
ma-10-1-5	41.75	0.52 / 338.04	47.56	1 / 385
ma-10-1-10	53.92	0.95 / 350.07	78.1	1 / 426
ma-10-1-15	64.61	0.93 / 397.89	80.28	1 / 519
ma-10-2-5	68.21	3.5 / 377.52	78.95	4 / 444
ma-10-2-10	93.17	4.17 / 617.31	112.12	5 / 749
ma-10-2-15	112.56	3.22 / 705.74	143.62	4 / 945
ma-10-3-5	132.55	12.34 / 434.68	156.0	13 / 527
ma-10-3-10	135.54	9.53 / 668.54	166.3	11 / 877
ma-10-3-15	160.32	6.27 / 772.98	202.26	7 / 907
ma-10-4-5	165.31	10.99 / 817.93	193.68	12 / 960
ma-10-4-10	191.31	10.42 / 877.08	231.72	12 / 1050
ma-10-4-15	239.05	13.37 / 628.09	280.18	15 / 773
ma-10-5-5	255.41	27.44 / 730.04	301.82	33 / 885
ma-10-5-10	310.90	22.07 / 800.67	385.58	27 / 960
ma-10-5-15	282.47	16.5 / 878.35	358.92	20 / 1162
ma-10-6-5	291.59	29.02 / 697.70	350.1	36 / 828
ma-10-6-10	320.61	71.90 / 978.49	325.58	59 / 945
ma-10-6-15	351.49	37.79 / 1021.05	366.84	39 / 1038
ma-10-7-5	374.32	127.78 / 1077.98	380.32	127 / 980
ma-10-7-10	429.75	61.48 / 1050.73	442.54	77 / 1077
ma-10-7-15	491.94	82.75 / 1531.86	498.02	88 / 1658

configurations where some agent(s) ultimately became trapped. Consequently, for each problem, the average, minimum and maximum figures were based on those 50 configurations where all agents could in principle and, actually did, finish successfully.

Brenner and Nebel (2009) stated that they placed a 10-minute time limit on their runs, counting a run as successful if all agents reached their respective goal locations within that time and unsuccessful otherwise. They then normalized the success rate relative to the success rate for the version where all agents had perfect knowledge of the entire grid occupancy – the version that invariably gave the best performance. Thus we cannot infer what absolute success rates were attained.[1] But, since the agents in their runs that could see only the immediately adjacent locations succeeded at a rate of only 37% - 62% *relative to* the full visibility case (and thus at a lower rate *absolutely*), it is safe to say that many of their runs failed within the 10 minute time bound. Their focus apparently was less on run times, and more on showing that a certain amount of forgetfulness (e.g., only remembering information gained over the last 5 steps) is helpful when the world is imperfectly known, presumably because remembering locations of other agents is deceptive – they also are in motion.

If we assume that the one example Brenner and Nebel display (a 6 by 6 grid with 4 agents and 10 blocked locations) is somewhat typical, then our run times fall far below their 10-minute time limit. In fact, for that particular world, ma-6-4-10 in Table 7.4, our agents finished on average in 15.63 seconds (and the worst time was 63.10 seconds); and even for a 10 by 10 grid with up to 7 agents, our average run times were well under the 10 minute mark. Our results are thus surprisingly good, especially since the planning technique employed in [15] is quite sophisticated, somewhat resembling Hierarchical Task Network planning (as discussed towards the end of that paper).

Having shown that our agent framework has the flexibility to hold its own against recent continuous planners, we should reiterate that our framework provides a range of features beyond simple planning, including a degree of self awareness and a question-answering ability.

[1]Our efforts to obtain this information were unsuccessful.

Chapter 8

Related Work on Agent Systems

We now relate our agent framework to (noncommunicative) cognitive robots and behavioral agents, and to previous agent systems that combine planning, reasoning and dialogue to some degree. We also discuss some general architectural proposals for autonomy and self-awareness.

8.1 Cognitive robots

First we consider the area of *cognitive robotics* (e.g., [47, 31, 69]), a term used by many to distinguish robots that reason symbolically from ones more focused on proficient motion or manipulation (see Section 8.2). Foundational work in this area has addressed challenging issues in planning, including conditional and sensing actions, iteration, incomplete knowledge, qualitative user preferences, uncertainty or nondeterminism, exogenous events, and reasoning about the planning agent's own mental states, to be anticipated as a result of contemplated actions. A noteworthy tool in some of this work has been the Golog agent programming language [67], which layers formal procedural constructs on top of the Situation Calculus.

However, in practice cognitive robots have mostly (though not exclusively) been concerned with motion planning aimed at externally supplied goals. While they allow for incomplete advance knowledge of the disposition of obstacles and target objects, there is typically no significant capacity for general reasoning about mundane objects, or about their own or other agents' mental states.

The earliest example of a cognitive robot was Shakey [62, 61], which used the STRIPS planner and resolution theorem proving to solve problems involving navigation through various rooms and moving around boxes; it could also explain its actions (by reference to its goal stack). Examples of later cognitive robots, with emphasis on navigation and manipulation planning, were Diablo [8], RHINO [17], DORO [18], ARMAR [43, 64], and several SplinterBot mobile robots programmed as cognitive robots (e.g., [45]). Several of these robots (Diablo, DORO, and the Splinterbot robots at CMU) were concerned with "tidying-up" (of colored blocks) or "rescue" tasks; all performed symbolic motion planning, and several augmented their map knowledge as they moved about. RHINO, a "museum tour guide", was impressive in its ability to deal with unpredictably moving humans, communicate its current goals through a display, and appropriately play prerecorded messages. A particularly interesting recent robot is Leonardo [14], which can perform certain cooperative tasks and rule-learning tasks in table-top worlds of children's blocks, boxes, and visual barriers. This engaging robot uses propositional STRIPS-like goal-directed schemas, accumulates quasi-symbolic, time-stepped trajectory knowledge about objects (e.g., (block, red, triangle)) it perceives with the help of reflective markers, and also ascribes goal schemas and trajectory knowledge to its human collaborator. It has been used for mind modeling experiments and for learning rules such as "put a marble on each blue block and each red block" based on an automated or human demonstration. In terms of behavior and elementary mind-modeling, Leonardo is perhaps the most interesting cognitive robot, but its knowledge representations are narrower in scope than those in Golog-based or other logic-based robots.

The capabilities of these robots evidently intersect those of agents in our framework, and many of them address issues such as uncertainty, physical perception, and behavioral learning that our framework currently excludes. However, robot development requires heavy investment in the design of perceptual and motion capabilities. This leaves less scope for developing more general reasoning or planning capabilities, such as are needed in modeling a world containing various types of objects with various types of properties and uses, and agents whose beliefs, wants and intentions need to be taken into account.

8.2 Behavioral Agents

A "behavioral" agent is one whose activities are determined by immediate environmental features and the agent's own current state. Their relevance to our enterprise lies in their reactivity, in the sense that they exploit opportunities or avoid threats as they arise. However, they generally lack symbolic reasoning and planning, apart from, perhaps, exploratory look-ahead in the state space. We have already noted that reinforcement learning (RL) agents are of this type. While RL agents learn a behavioral policy as a result of extensive exploration and the concomitant reward/punishment experience, such a policy is simply a way of reacting to the current state. The disadvantage of this way of coping with an environment containing risks and opportunities is that it does not scale up well to complex worlds (one in which a wide variety of entities and situations may be encountered, so that the number of successors of a state can be extremely large), and that learned policies are very domain-specific. The advantage of deliberation – symbolic thinking – is precisely its generality.

Autonomous agents in game worlds are typically programmed as purely behavioral agents. Some may be endowed with a planning capability, in some limited sense. For example, agents described in [25] are able to choose a motion trajectory as a function of features of the current situation (where the motion sequences are learned by segment extraction from demonstrations by a "teacher"). The camera-equipped table-soccer machine KiRo [77], like some game agents considers multiple action sequences it might carry out (or that might occur if the opponent gains control of the ball), and evaluates their expected utility before choosing an action. Some game agents (e.g., [23]) may make use of ideas from the intelligent agent literature (such as the BDI model – see Section 8.4), but in practice remain rule-based (*navigate to a health pack when in diminished health, attack enemies when in good health, etc.*). Techniques tend to be very domain-specific, with no provision for cognitive, let alone reflective, capabilities or intelligent verbal interaction (but see [51], discussed in Section 8.3).

Most humanoid robots, despite their life-like features, are also purely behavioral, and as such incapable of using or acquiring symbolic knowledge. For example, the control structure of MIT's Kismet robot [13], a predecessor of the Leonardo robot, consists of a hierarchy of IF-THEN rule sets, where the IF-part of each rule tests for perceptual inputs and internal state parameters, and the THEN-part selects a successor rule set, where the rule sets bottom out in modes of behavior such as looking happy or sad, turning towards or away from a person or object, or making sounds to attract attention. The humanoid robots in research labs and commercial enterprises worldwide are too numerous to mention, but while there are some moves towards higher cognitive functions, the main emphasis continues to be on improved perceptual and motion capabilities, while interactions with objects and humans is preprogrammed and, in the case of verbal interaction, scripted.

8.3 Agents that Plan, and Communicate Verbally

Several agent systems and frameworks more nearly resemble ours than any of the planning systems, cognitive robotics projects, or behavioral agent systems mentioned above.

A remarkable early system was Winograd's SHRDLU [84], which not only was able to plan and act in a simulated world of children's blocks on a table, but did so while taking instructions and answering questions in English via a keyboard interface. Its question-answering and planning capabilities within its blocks domain were quite advanced, thanks to its proceduralized linguistic and planning knowledge. Unfortunately the complexity and lack of transparency of the procedural representations, and the subtlety of procedural interactions, seems to have forestalled follow-up work targeting broader or more ambitious domains.

An even more impressive agent with planning and dialogue capabilities was Vere and Bickmore's Homer [80], a small simulated robotic submarine navigating a 2-dimensional Seaworld. The agent could form and execute plans to navigate to particular places, pick up and deliver parcels, "take pictures" of objects in passing, and infer larger-scale event types (such as passing an object) from multiple smaller-scale ones (such as having the object ahead of it at one point and behind it at another). Like SHRDLU, it could take instructions and report its plans, actions and observations in English, but used a larger vocabulary (about 800 words), and showed a somewhat greater degree of self-awareness.

Compared with SHRDLU's and Homer's sophisticated goal-directed planning in geometrical worlds, and their advanced English capabilities, our Gridworld agents are certainly less fully developed. However, our project was never concerned with engineering an elaborate simulation nor, at this point, complex linguistic skills, but rather with providing a flexible framework for defining self-motivated agents that can effectively reason about their world (including themselves), can use their knowledge to plan for long-term cumulative rewards, and can answer simple questions about their world and themselves. Thus we (as well as some undergraduates in our courses) have been able to implement various kinds of self-motivated agents and even classical planners and multiagent worlds (as seen above) in our framework. The English QA capabilities, though elementary in form, allow for inference, cover both "physical" properties and mental attitudes, and are easy to extend to new domains. Though SHRDLU and Homer were important AI milestones, their domain-specific procedural aspects (in both planning and dialogue) shackled them to their particular domains, and forfeited a property that (as John McCarthy maintained) seems crucial for scaling up AI: elaboration-tolerance.

Two more recent, highly developed agent projects that overlap conceptually with ours the GLAIR/Cassie project [73, 72], and the TRIPS project [26]. Cassie, as a virtual agent or as a Nomad robot, operated in a world of large colored cubes and balls, interpreted as other robots and two people. It could follow commands such as "Go to the green robot" and "Follow Bill", commented on its actions by saying, for example, "I found Bill. I went to Bill. I am near Bill. I am following Bill", and it could answer questions such as "Who are you?", and "Who have you talked to?" (thus exhibiting basic self-awareness). Its planning relied on formal rules such as that if an agent is to follow an object, then it needs to find the object, so as to be looking at it, and then go to it so as to be near it. The acting executive described in [73] performs goal-regression, though the various versions of Cassie seem to have relied mostly on predefined plans, and on policies about when certain acts should be performed. The knowledge representation is partially equivalent to FOL, but allows (unquoted) formulas as terms, leading to an approach to modals not unlike ours. The GLAIR architecture is more ambitious than ours, in its allowance for perceptual and motor layers beneath the conscious level, and Cassie features an episodic memory allowing QA about past interactions. The latter remains an unimplemented goal in our Gridworld framework, though our group has previously demonstrated episodic memory in a related QA system focused on self-awareness (see [70], which draws on earlier work by [41]; see also [58]). Distinctive characteristics of our framework again include the forward-directed, reward-optimizing planning, the systematic elaboration of world descriptions through forward inference, and the use of the restricted CWA to allow uniform handling of properties of objects at hand (or at a remove) and of attitudes, for both inference and QA.

The TRIPS system [26] is the result of many years of development. It stands out as an end-to-end system that not only performs problem solving and planning in a fairly complex simulated world, but does so collaboratively, interacting with human users via spoken language. A display shows the island and routes being followed, highlights salient entities, shows the speech recognizer output, and allows for certain GUI interactions. The planner is geared towards the evacuation domain, but many operations relating to the construction of a shared plan, such as proposing, accepting, or rejecting potential actions, modifying aspects of actions, and retracting parts of plans, are relatively domain-independent at an abstract level. The use of generic components has allowed Allen's research group to rather quickly develop new prototype applications, such as patient health monitoring [27], and collaborative learning of web tasks [39]. Of course, nobody yet knows how to achieve broad-coverage, domain-independent language understanding, so TRIPS employs production-system-like heuristic rules to map preliminary logical forms derived from parse trees into plausible intentions underlying the user's utterances. The intentions are typically to propose an addition to, or alteration of, the current plan, or to ask a question.

Our Gridworld framework certainly lacks TRIPS's focused collaborative problem-solving and linguistic skills and well-developed virtual and prototype application domains and interfaces. But it has the distinctive features we have pointed out before: It makes uniform inferential use of general, logically represented knowledge for planning and QA (while reasoning in TRIPS consists primarily of instantiating recipes that promise to achieve some current objectives), allows for reasoning about its own and other agents' attitudes, and acts to try to maximize its own future rewards. Also the transparent declarative form in which knowledge and plans are represented make instantiation of new agents quite simple.

Another interesting recent line of work tackles agent design with formal tools that include a temporal action logic (TAL) equipped with natural deduction rules for planning and QA, and an incremental chart parser for a small fragment of English that offers a few alternative continuations to the user after each word that is typed in [51, 52]. TAL is extended so as to allow representation of beliefs, inform acts and request acts using quoted sentences. The first of two systems briefly described in these papers is a quest game scenario where the user-controlled avatar encounters two characters from whom he makes a profit by buying lumber from one and selling it to the other. The avatar and two characters interact linguistically and plan and act, answering questions about what objects they own and their prices, offering to buy or sell objects (and doing so), moving to a new location, chopping down trees, and reporting on locations and "what happened". The second system is concerned with a search and rescue scenario for an autonomous helicopter. The agent makes such plans as flying to a cell (on a grid), scanning it, and reporting the body count detected by the scan; the latter may require requesting another mobile agent to relay the report.

This work is impressive in its use of the extended TAL (which also involves use of occlusion for nonmonotonic reasoning), and the implementation of interacting agents capable of simple planning (including replanning) and physical and speech acts in their worlds. The use of quotation is similar to our use of reification operators, and as in our system, there is limited forward inference as well as deductive QA. Some differences from our work are that the agents are given simple goals (such as possession of lumber) rather than being self-motivated in our sense. The handling of agent interactions via utterances interpreted as speech acts is more advanced than in our framework; on the other hand our framework allows for questions that are explicitly about knowing, knowing-whether or knowing-that.

Perhaps most importantly, our approach to planning based on STRIPS-like operators and a partial CWA seems to enable implementation of considerably more complex scenarios (such as our main example as well as the classical and continual planning examples) than the TAL-based approach.

8.4 Some General Architectures for Intelligent Autonomous Systems

Finally, we comment on the relationship between our agent framework and some well-known architectural proposals for intelligent agents. One such proposal is the belief-desire-intention (BDI) architecture of [12], which emphasizes the pragmatic advantages of committing to a plan, thereby limiting the possible futures that need to be thought about by an agent in its effort to fulfill its desires. Motivation in our agents is in a sense more fundamental than in BDI agents: Rather than taking desires to be primary, they ground desires (and along with them, plans and intentions) in a system of rewards and penalties. Thus our agents may be deflected from their current plan of action by new opportunities and threats, even if the current plan still seems successfully executable.

Various types of *decision-theoretic* agent architectures (e.g., [71]) model agent behavior as a mapping from percepts and internal states to actions, where – much as in our conception – the goal is to optimize the utility of an agent's history, as determined by the values (to the agent) of the environmental states reached. While in principle such a perspective allows for arbitrary internal processing by the agent (limited only by its programming language, its software and hardware architecture, and the need to act promptly), it tends to promote behaviorist rather than cognitivist approaches to agent construction. Typically problems similar to those we considered in Sections 7.2 and 7.3 are addressed (often allowing for stochastic environments), and techniques such as metacontrol, intention reconsideration, and POMDPs are employed. But in our view, the decision-theoretic aspects of action selection are ultimately far less important in intelligent behavior than the knowledge on which the decisions are based. We do not know of any decision-theoretic agents that plan to achieve high utilities, and reason and answer questions about the world and themselves, at the level of our agents.

Several architectural frameworks bear the imprint of Allen Newell's cognitive perspective [60]. One is Soar [44], a programming environment for building rule-based expert systems, equipped with a working memory, semantic memory, and episodic memory (as in the SplinterBot cognitive robot mentioned above). As an agent architecture, Soar lends itself well to carrying out multiple tasks in time-shared fashion, and thus has been used in various simulation-like studies for single and multiple agents, such as collision avoidance for autonomous air vehicles, or simulation of events such as fire, smoke, and flooding on ships in a naval fleet. The architecture seems less well-suited to complex reasoning, planning, and dialogue, in part because productions – like the procedural knowledge employed in SHRDLU and other early agents – are apt to encode knowledge in a rather opaque way, and in part because the knowledge representations traditionally used in semantic and episodic memory are generally limited to simple attribute-value constructs, with no systematic allowance for logical connectives, quantifiers, or semantically well-founded representations of attitudes and other modalities. Thus the kind of systematic, reasoned lookahead and QA performed by agents in our framework seems to us difficult to achieve within the Soar architecture.

Another architecture partially indebted to Newell is ACT-R [4], the product of a lengthy cognitive modeling tradition. The emphasis in this architecture is on simulating human problem-solving and learning in empirically testable detail. Interesting models of performance on various memory tasks, learning tasks and problem solving tasks have been implemented within the ACT-R framework, and in recent years there has also been work on designing cognitive ACT-R-based agents. For example, [10] describe initial work on virtual agents intended to act as virtual opponents to military trainees in urban combat. Productions specify actions such as shooting at or evading an enemy as a function of available escape routes, current goals, and the agent's assigned role. However, the agents are constrained by the coordinated behaviors scripted by the programmers and by the goals assigned to them. They do not think ahead, do not make knowledge-based inferences, and cannot in any genuine sense consider and answer questions.

Chapter 9

Conclusion and Future Work

Our framework has demonstrated the feasibility of combining planning, inference and inferential question answering in a completely self-motivated cognitive agent. ME, as demonstrated in our main example, plans deliberately and continuously, and acts opportunistically in accord with its reasoned expectations about future rewards and penalties. It does so by drawing on its specific and general knowledge about itself and its environment, including both introspective (autoepistemic) knowledge and knowledge about mental states of other agents. Additionally, ME can perform bounded forward inference using its current knowledge in a given state. We also demonstrated the flexibility of our utility-based planning technique by tackling some classical goal-directed planning problems, and recently proposed problems in continual goal-directed planning by single or multiple agents in incompletely known worlds, showing that our approach is competitive in these areas.

Several extensions to the Gridworld framework would be desirable. One is allowance for degrees of uncertainty about the truth of some predications and about the effects of actions. Thus in exploring future states, not only alternative actions, but also alternative outcomes of those actions (with probabilities) would be considered, and the plans formed would generally be *conditional* (contingency) plans. Also, if ME is uncertain whether the preconditions of an action hold, it should nonetheless consider attempting that action, especially if the potential rewards of success are high and the risks are low.

Conditional occlusion is another desirable enhancement; e.g., what is contained in a box might be occluded if the lid is closed, but unoccluded if the lid is open. Instead of marking predicates as definitely occluded, we would like to replace the notion of occlusion with explicit metaknowledge. For example, the agent might have a metacognitive axiom to the effect that if it can see the interior of a container and does not see a particular item in it, then that item is not in the container (cf. [58]).

A capacity for learning would be another desirable extension of the Gridworld framework. This could be effected by experience-based modification of rewards for certain actions or states that satisfy the preconditions of other, particularly rewarding actions. This would be a form of reinforcement learning, akin to the joy of anticipation (or dread) that humans (and other creatures, such as Pavlov's dog) can develop as a result of experience. Abstraction of macro-operators from particularly rewarding sequences of actions (much as in the tradition of [29]) is another possibility.

Our long-term vision is that of an explicitly self-aware and self-motivated conversation agent, communicating about the real world, not a simulated one, and perhaps at a later stage equipped with perceptual and motor abilities. Work on plan-based dialogue has been somewhat neglected over the past 20 years (since its beginning with [19] and [3]), but we believe that our ideas about rapidly adaptable continuous planning based on efforts to secure cumulative rewards and avoid negative consequences can contribute usefully to this area. Planning would be driven by general ideas about the effects and value of dialogue acts (at multiple levels of description) and by noticing the relevance of certain dialogue acts in the current situation. Forward inference would be used to predict indirect consequences of dialogue acts, including additions to the common ground, and expected rewards and risks. Rewards in this case would be, for example, gaining desired information, approval or other forms of "social capital", or more concrete help or gains; negative consequences are also easily enumerated.

Another feature of Gridworld that seems transferable to future cognitive systems is the method of reflective inference employed by the system and ascribed to other agents. The idea that agents know what they know and don't know is of course familiar, but that an agent X knows the truth or falsity of broad classes of statements whose subject is X is less familiar. While this assumption cannot be made about all predicates (e.g., whether X hosts certain microorganisms, whether X has read more novels than her neighbor, or will die within 20 years), it can be made about most attitudes (what X wants, likes, fears, hopes, expects, intends, doubts, etc.), about major abilities (what devices X can operate, what languages X can speak, what foods X can prepare, etc.), about ownership, friendship, acquaintanceship, major life events, and much more.

Bibliography

[1] A. Albore, H. Palacios, and H. Geffner. A translation-based approach to contingent planning. In *Proceedings of the 21st international Joint Conference on Artifical intelligence (IJCAI 2009)*, pages 1623–1628, 2009.

[2] J. F. Allen, N. Chambers, G. Ferguson, L. Galescu, H. Jung, M. Swift, and W. Taysom. PLOW:a collaborative task learning agent. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007)*, 2007.

[3] J.F. Allen and R. Perreault. A plan-based analysis of indirect speech acts. *Computational Linguistics*, 6(3-4):167–182, 1980.

[4] J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of mind. *Psychological Review*, 111:1036–1060, 2004.

[5] M. L. Anderson and D. R. Perlis. Logic, self-awareness and self-improvement: The metacognitive loop and the problem of brittleness. *Journal of Logic and Computation*, 15(1):21–40, 2005.

[6] A. Aydemir, M. Göbelbecker, M. Hanheide, N. Hawes, I. Kruijff-Korbayová, P. Lison, K. Sjöo, J.L. Wyatt, H. Zender, and M. Zillich. Dora the explorer: A mobile robot motivated by curiosity. In *poster, 4th Int. Conf. on Cognitive Systems (CogSys 2010)*, Zurich, Jan. 27–28 2010.

[7] F. Bacchus and R. Petrik. Modeling an agent's incomplete knowledge during planning and execution. In A.G. Cohn, L.K. Schubert, and S.C. Shapiro, editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR 1998)*, pages 432–443. Morgan Kaufmann Publishers, 1998.

[8] C. Baral, L. Floriano, A. Hardesty, D. Morales, M. Nogueira, and T. C. Son. From theory to practice: the utep robot in the aaai 96 and aaai 97 robot contests. In *Proc. of the 2nd Int. Conf. on Autonomous Agents (Agents-98)*, pages 32–38, St. Paul,MN, May 9-13 1998.

[9] C. Baral and T.C. Son. Approximate reasoning about actions in the presence of sensing and incomplete information. In *Proceedings of the International Logic Programming Symposium (ILPS 1997)*, pages 387–401, 1997.

[10] B.J. Best and C. Lebiere. Cognitive agents interacting in real and virtual worlds. In R. Sun, editor, *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*, pages 186–218. Cambridge Univ. Press, 2006.

[11] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS 2000)*, pages 52–61, 2000.

[12] M. E. Bratman. What is intention? In P. R. Cohen, J. L. Morgan, and M. E. Pollack, editors, *Intentions in Communication*, pages 15–32. MIT Press, 1990.

[13] C. Breazeal and R. Brooks. Robot emotion: A functional perspective. In J.-M. Fellous and M.A. Arbib, editors, *Who Needs Emotions: The Brain Meets the Robot*, page Chapter 10. Oxford Univ. Press, 2004.

[14] C. Breazeal, J. Gray, and M. Berlin. An embodied cognition approach to mindreading skills for socially intelligent robots. *The Int. J. of Robotics Research*, 28:565–680, May 2009.

[15] M. Brenner and B. Nebel. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems*, 19(3):297–331, December 2009.

[16] D. Bryce, S. Kambhampati, and D.E. Smith. Planning graph heuristics for belief space search. *Journal of Artificial Intelligence Research*, 26:35–99, 2006.

[17] W. Burgard, A. B. Cremers, D. Fox, D Hahnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55, 1999.

[18] A. Carbone, A. Finzi, A. Orlandini, F. Pirri, and G. Ugazio. Augmenting situation awareness via model-based control in rescue robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS-05)*, Edmonton, Canada, August 2–6 2005.

[19] P.R. Cohen and R. Perreault. Elements of a plan based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

[20] M. Cox and A. Raja. Metareasoning: A manifesto. In *AAAI-08 Workshop on Metareasoning*, pages 1–4, July 13-14 2008.

[21] M. T. Cox. Metacognition in computation: A selected research review. *Artificial Intelligence*, 169(2):104–141, 2005.

[22] M. T. Cox and A. Raja, editors. *Metareasoning: Thinking about Thinking (to appear)*. MIT Press, 2009.

[23] N.P. Davies and Q. Mehdi. BDI for intelligent agents in computer games. In *Proceedings of the 8th International Conference on Computer Games: AI and Mobile Systems (CGAIMS 2006)*, 2006.

[24] G. de Giacomo, L. Iocchi, D. Nardi, and R. Rosati. Moving a robot: the kr & r approach at work. In *Proc. of the 5th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 198–209, Cambridge, MA, November 5-8 1996.

[25] J. Dinerstein, P.K. Egbert, D. Ventura, and M. Goodrich. Demonstration-based behavior programming for embodied virtual agents. *Computational Intelligence*, 24(4):235–256, 2008.

[26] G. Ferguson and J. F. Allen. TRIPS: An integrated intelligent problem-solving assistant. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI 1998)*, 1998.

[27] G. Ferguson, J. Quinn, C. Horwitz, M. Swift, J. Allen, and L. Galescu. Towards a personal health management assistant. *J. of Biomedical Informatics*, 43(5):S13–S16, 2010.

[28] E. Ferrein, C. Fritz, and G. Lakemeyer. On-line decision-theoretic Golog for unpredictable domains. In *Proceedings of the 27th German Conference on Artificial Intelligence (KI 2004)*, 2004.

[29] R.E. Fikes, P.E. Hart, and N.J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[30] M. Fox and D. Long. PDDL 2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[31] C. Fritz and S.A. McIlraith. Planning in the face of frequent exogenous events. In *Proc. of the 1st Int. Symp. on Search Techniques in Artificial Intelligence and Robotics (at AAAI08)*, Chicago, IL, July 13–14 2008.

[32] J. Funge, X. Tu, and D. Terzopoulos. Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999.

[33] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1987.

[34] K. Golden, O. Etzioni, and D. Weld. Tractable closed-world reasoning with updates. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 178–189, 1994.

[35] Nick Hawes. A survey of motivation frameworks for intelligent systems. *Artificial Intelligence*, 175(5-6):1020–1036, 2011.

[36] B. Hayes-Roth and F. Hayes-Roth. A cognitive model of planning. *Cognitive Science*, 3:275–310, 1979.

[37] J. Hoffman and R. Brafman. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS 2005)*, pages 71–80, 2005.

[38] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[39] H. Jung, J. Allen, L. Galescu, N. Chambers, M. Swift, and W. Taysom. Utilizing natural language for one-shot task learning. *J. of Logic and Computation*, 18(3):475–493, 2008.

[40] A. N. Kaplan and L. K. Schubert. A computational model of belief. *Artificial Intelligence*, 120(1):119–160, 2000.

[41] A.N. Kaplan. Reason maintenance in a hybrid reasoning system. In *Proc. of the 1st Workshop in Computational Semantics (ICoS-1)*, 1999.

[42] H. A. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI 1992)*, 1992.

[43] N. Krüger, J. Piater, C. Geib, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrcen, A. Agostini, and R. Dillmann. Object-action complexes: Grounded abstractions of sensory-motor processes. *Robotics and Autonomous Systems*, 59(10):740–757, 2011.

[44] J. E. Laird, A. Newell, and P. S. Rosenbloom. SOAR: an architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[45] J.E. Laird, N. Derbinsky, and J. Voigt. Performance evaluation of declarative memory systems in soar. In *Proc. of the 20th Behavior Representation in Modeling & Simulation Conference*, pages 33–40, Sundance, UT, March 21–24 2011.

[46] Pat Langley. Cognitive architectures and general intelligent systems. *AI Magazine*, 27(2):33–44, 2006.

[47] H. Levesque and G. Lakemeyer. Cognitive robotics. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, page Chapter 24. Elsevier, 2008.

[48] D. H. Liu. A survey of planning in intelligent agents: from externally motivated to internally motivated systems. Technical Report TR-2008-936, Department of Computer Science, University of Rochester, 2008.

[49] D. H. Liu and L. K. Schubert. Incorporating planning and reasoning into a self-motivated, communicative agent. In *Proceedings of the 2nd Conference on Artificial General Intelligence (AGI 2009)*, 2009.

[50] D. H. Liu and L. K. Schubert. Combining self-motivation with logical planning and inference in a reward-seeking agent. In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence (ICAART 2010)*, volume 2, pages 257–263, 2010.

[51] M. Magnusson and P. Doherty. Temporal action logic for question answering in an adventure game. In P. Wang, B. Goertzel, and S. Franklin, editors, *First Conf. on Artificial General Intelligence (AGI-08), vol. 171 of Frontiers in Artificial Intelligence and Applications*, pages 236–247. IOS Press, February 2008.

[52] M. Magnusson, D. Landén, and P. Doherty. Logical agents that plan, execute, and monitor communication. In *2nd Workshop on Logic and the Simulation of Interaction and Reasoning (LSIR-2)*, Pasadena, CA, July 2009.

[53] J. McCarthy. Making robots conscious of their mental states. In *Machine Intelligence 15*, pages 3–17, 1995.

[54] Drew McDermott. *The planning domain definition language manual.* 1998.

[55] K. Merricka and K. Shafib. Agent models for self-motivated home-assistant bots. In *Int. Symp. on Computational Models for Life Sciences (CMLS-09)*, pages 131–150, 2009.

[56] R. Micalizio. A distributed control loop for autonomous recovery in a multi-agent plan. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2009.

[57] F. Morbini and L. Schubert. Towards realistic autocognitive inference. In *Logical Formalizations of Commonsense Reasoning, Papers from the AAAI Spring Symposium*, pages 114–118. AAAI Press, March 26-28 2007.

[58] F. Morbini and L. Schubert. Metareasoning as an integral part of commonsense and autocognitive reasoning. In *AAAI-08 Workshop on Metareasoning*, 2008.

[59] F. Morbini and L.K. Schubert. Conscious agents. Technical Report TR-2005-879, Department of Computer Science, University of Rochester, 2005.

[60] Allen Newell. *Unified Theories of Cognition.* Harvard University Press, Cambridge, MA, 1990.

[61] N. J. Nilsson. Shakey the robot. Technical Report 323, AI Center, SRI International, 1984.

[62] R.E. Fikes N.J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[63] H. Palacios and H. Geffner. From conformant into classical planning: Efficient translations that may be complete too. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS 2007)*, pages 264–271, 2007.

[64] R. Petrick, N. Adermann, T. Asfour, M. Steedman, and R. Dillmann. Connecting knowledge-level planning and task execution on a humanoid robot using Object-Action Complexes. In *poster, 4th Int. Conf. on Cognitive Systems (CogSys 2010)*, Zurich, Jan. 27–28 2010.

[65] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann Publishers, 1991.

[66] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems*, 1995.

[67] R. Reiter, editor. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, Cambridge, MA, 2001.

[68] E. D. Sacerdoti. A structure for plans and behavior. Technical Report 109, AI Center, SRI International, 1975.

[69] S. Sardiña, G. De Giacomo, Y. Lespérance, and H. Levesque. On ability to autonomously execute agent programs with sensing. In *Proc. of the 4th Int. Workshop on Cognitive Robotics (CoRobo-04)*, 2004.

[70] L. Schubert. Some knowledge representation and reasoning requirements for self-awareness. In M. Anderson and T. Oates, editors, *Metacognition in Computation: Papers from the 2005 AAAI Spring Symposium*, pages 106–113. AAAI Press, 2005.

[71] M. Schut and M. Wooldridge. The control of reasoning in resource-bounded agents. *Knowledge Engineering Review*, 16(3):215–240, 2001.

[72] S.C. Shapiro. Embodied Cassie. In *Cognitive Robotics: Papers from the 1998 AAAI Fall Symposium*, page 136143, Menlo Park, CA, 1998.

[73] S.C. Shapiro and J.P. Bona. The GLAIR cognitive architecture. *International Journal of Machine Consciousness*, 2(2):307–332, 2010.

[74] S. Singh, D. Litman, M. Kearns, and M. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

[75] S. P. Singh, A. G. Barto, R. Grupen, and C. Connolly. Robust reinforcement learning in motion planning. In *Advances in Neural Information Processing Systems 6*, pages 655–662. Morgan Kaufmann, 1994.

[76] D. Sonntag. Introspection and adaptable model integration for dialogue-based question answering. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2009.

[77] M. Tacke, T. Weigel, and B. Nebel. Decision-theoretic planning for playing table soccer. In *Proceedings of the 27th German Conference on Artificial Intelligence (KI 2004)*, 2004.

[78] M. Tambe, W. L. Johnson, O. M. Jones, F. K., J. E. Laird, P. S. Rosenbloom, and K. Schwamb. Intelligent agents for interactive simulation environments. *AI Magazine*, 16:15–39, 1995.

[79] M.C. Torrance. Natural communication with robots. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, 1994.

[80] S. Vere and T. Bickmore. A basic agent. *Computational Intelligence*, 6(1):41–60, 1990.

[81] M. A. Walker. An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email. *Journal of Artificial Intelligence Research*, 12:387–416, 2000.

[82] D.S. Weld. Recent advances in AI planning. *AI Magazine*, 20(2):93–123, 1999.

[83] T. Winograd. Procedures as a representation for data in a computer program for understanding natural language. Technical Report 235, Massachusetts Institute of Technology, 1971.

[84] T. Winograd. *Understanding Natural Language.* Academic Press, 1972.