COMBINING SELF-MOTIVATION WITH LOGICAL PLANNING AND INFERENCE IN A REWARD-SEEKING AGENT

Daphne Liu and Lenhart Schubert*
Department of Computer Science, University of Rochester, Rochester, NY, U.S.A.

Keywords: Self-aware agent, Self-motivated cognitive agent, Introspection.

Abstract: We present our preliminary work on our framework for building self-motivated, self-aware agents that plan 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 quest for rewards and their continual plan updates. The framework allows for both specific and general (quantified) knowledge, epistemic predicates such as *knowing-that* and *knowing-whether*, for incomplete knowledge of the world, for quantitative change, for exogenous events, and for dialogue actions. Question answering and experimental runs are shown for a particular agent *ME* in a simple world of roads, various objects, and another agent, demonstrating the value of continual, deliberate, reward-driven planning.

1 INTRODUCTION

Our interest is ultimately in intelligent, linguistically competent agents. Such agents must be capable of planning in an incompletely known world, inference over a substantial knowledge base (including a self-model), dialogue, and unceasing pursuit of their own rewards. Many AI systems have demonstrated *some* of these abilities, but none to our knowledge have integrated them all. We present a simple framework for defining such self-motivated cognitive agents in simulated worlds and exemplify this with an agent dubbed *ME* (for Motivated Explorer). Future work will move the domain to the real world (rather than more elaborate simulated worlds) and emphasize dialogue. *ME* can plan and think ahead (where actions produce gradual change over time), deal with unforeseen events, make inferences about states (including its own and other agents' knowledge and wants), engage in 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 the type of self-motivated agent described here, traditional planners have been constrained to act and react in accord with user-specified goals, constraints and preferences. Self-motivation would seem to lead to single-minded, self-indulgent behavior, but this is not at all so. If the agent "enjoys" acquiring knowledge, exploring its environment, conversing, and satisfying perceived user goals, then it can appear curious, interactive, and helpful. It may also seem that such an agent could do no more than typical reinforcement learning (RL) agents. In fact, the kind of agent envisaged and illustrated here conceives of its world and its actions in logical descriptive terms – it is a *cognitive* agent and thus able to plan ahead, entertain beliefs about itself and other agents, and engage in question-answering dialogue.

In Section 2, we describe what we mean by a self-motivated cognitive agent, and our development framework for implementing such agents. In Section 3 we detail 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 Section 4 we focus on the reasoning (including introspection) performed by the agent.

In Section 5, we instantiate the agent *ME* in a particular simulated world in order to demonstrate and experiment with our framework. The simple sample world suffices for illustrating and supporting the points we wish to make. In Section 6 we then provide examples of question-answering by *ME*, and re-

*This work was supported by NSF grants IIS-0535105 and IIS-0916599.

Liu D. and Schubert L. (2010).
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 - Agents*, pages 257-263
DOI: 10.5220/0002700602570263

port results of various "runs" of the agent in the simulated world, showing the advantages for the agent of its deliberate, self-motivated style of continual planning. We conclude with a summary, a contrast with related work, and future directions.

2 SELF-MOTIVATED COGNITIVE AGENT FRAMEWORK

We conceive of a self-motivated cognitive agent as one whose activity is governed by continual reasoned elaboration and execution of a life plan (as suggested in (Morbini and Schubert, 2008); see also (Liu and Schubert, 2009)). It is not assumed to have complete knowledge either about the current world state or about exogenous events, but it observes new or altered facts as it goes. Such an agent manifests *opportunistic* behavior; i.e., as new facts become apparent to it or actions fail, its continual planning will deal appropriately with unanticipated opportunities and hazards.

Accordingly, the framework and (Lisp-based) platform 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 depends on the exact action parameters involved and on arbitrary descriptive or numerical features of the world and the agent. Actions and states with large expected 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".

To be *cognitive*, an agent should plan and do so within an expressively rich language for describing, and making inferences about, actions and the world. We might have chosen a situation calculus-based framework like Golog (e.g., (Ferrein et al., 2004)), but both for ease of use in implementing a set of operators and for planning efficiency we chose a more STRIPS-like representation of actions and states. The action representation allows for quantitative preconditions and effects, handled by a general procedural attachment syntax. The logic of state descriptions also allows for propositional attitudes such as *knowing-that*, *knowing-whether* and *wanting*. This is essential for formalizing knowledge-acquisition actions and for answering questions about the agent's own attitudes.

The current framework confines *ME* to a simulated grid world. This consists of an arbitrary user-defined set of named locations, and named roads connecting some or all of these locations. Animate and inanimate entities of specified types can be placed at various locations. For animate entities (agents), the specified ground literals may include ones like beliefs and preferences. The types assigned to entities are separately specified, allowing shared properties of entities of that type to be listed. While *ME* is mobile and exploratory, interacting with the user and other entities, all other entities are stationary and merely reactive in their interactions with *ME*.

Aside from the general knowledge implicit in type specifications, grid worlds allow for specification of arbitrary conditional knowledge, such as that every sasquatch has an IQ of 50, or that if someone knows *p* is true then *p* is true. These general facts are stated in Horn clause-like form, with the important generalizations that the antecedent conditions of the clause may be positive or negative literals, and literals may be based on attitudinal predicates like *knows*, with reified arguments like $(that\ (is\_edible\ fruit3))$. These clauses are used for bounded forward inference in elaborating world state descriptions.

ME's knowledge base is 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. *ME*'s knowledge is incomplete even about entities at its current location. In particular, certain predicates are marked as being *occluded* for *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 *box*1. Similarly *knows* would generally be occluded, so that what another agent knows is not immediately apparent to *ME*. However, self-referential facts about *ME* such as $(has\ ME\ key1)$ and $(not\ (knows\ ME\ (whether\ (is\_edible\ fruit3))))$ are evident to *ME* (and thus added to *ME*'s world model), despite the general occlusion of *has* and *knows*. Also, *ME* may find out and remember a fact that would otherwise be occluded, perhaps because it asked a question, read a note containing the fact, or inferred it.

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 doesn't know.

3 ACTIONS, PLANNING, AND SIMULATION

Operators, or action types, 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. Consider the following *drink* operator with formal parameters ?*h* for *ME*'s thirst level, ?*x* for the item to be drunk, and ?*y* for the location of the item. From *ME*'s perspective, if it is thirsty and at the same location as a potable item, knows whether the item is potable, and there is no adverse environmental event, then *ME* can drink the item for 1 time unit and completely relieve its thirst.

```
(setq drink
  (make-op :name 'drink :pars '(?h ?x ?y)
  :preconds '((is_thirsty_to_degree ME ?h)
              (> ?h 0) (is_at ME ?y)
              (is_at ?x ?y) (potable ?x)
              (knows ME (whether (potable ?x)))
              (not (there_is_adversity)))
  :effects '((is_thirsty_to_degree ME 0)
             (not (is_thirsty_to_degree ME ?h)))
  :time-required 1    :value 50))
```

A plan consists of a sequence of actions. An action is created by replacing the formal parameters of an operator with actual values obtained by unifying its preconditions with facts in *ME*'s knowledge base. An action can be created in a given state only if *ME* finds its preconditions to be true according to *ME*'s knowledge in that state; such an action is *applicable* though *ME* might not elect to perform it. *ME* plans 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. 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 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.

Since *ME*'s knowledge of the world is incomplete, the actual effects of its actions may diverge from the effects expected by the agent. For example, if we model traveling (from *ME*'s perspective) as a multi-step action, but also allow for spontaneous fires that bring travel to a halt, then *ME* may not reach its expected destination. Given this divergence between expectations and reality, we clearly need to distinguish between *ME*'s conception of its actions and the "actual" actions in the simulated world. Space does not permit the illustration of "actual" operators, but their syntax is much like that of *ME*'s operators, except that *preconds* and *effects* are replaced by *startconds*, *stopconds*, *deletes*, and *adds*.

In the simulation, actual actions and exogenous events may be tracked in parallel in unit time steps. An active actual action will continue for another time step iff none of its stop conditions as given by *stopconds* are true in the current world state. The action will terminate immediately if any one of them is true in the current world state. 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.

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

The uniform procedural attachment technique is used in both preconditions and effects of actions. It enables both quantitative reasoning and dialogue. Some action preconditions and effects may contain evaluable terms (those whose operators are among $+$, $-$, $*$, $/$, $<$, $<=$, $=$, $>=$, $>$, *random*, and function names ending in ?). The system will evaluate these terms when verifying preconditions and when applying effects. 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. Similarly $(knows\ USER\ (that\ (answer\_to\_ynq?\ ?q)))$, an effect of *answer_ynq*, will have $(answer\_to\_ynq?\ ?q)$ evaluated as an answer formula when applied; the result might for example be $(knows\ USER\ (that\ (not\ (can\_fly\ guru))))$.

4 REASONING ABOUT WORLD STATES AND MENTAL STATES

Since *ME* does not generally know all the current facts, it cannot make full use of the CWA (unlike

many traditional planners). But we do not want to abandon the CWA altogether, as it would be inefficient to enumerate and maintain all the negative predications that hold even in simple worlds. Thus, *ME* uses the CWA only for (non-epistemic) predications in which *ME* is the subject of the predication. In this respect its self-knowledge is assumed to be complete. But when the predication concerns a non-*ME* subject, *ME* applies the CWA only for (1) predications about road connectivity and navigability (and even here the CWA could easily be weakened), and (2) (a) when the subject is a local entity currently colocated with *ME* or one *ME* has visited, and (b) the predication is non-occluded. In all other cases, the mere absence of a predication from the world model is not interpreted as supporting its negation – its truth value may simply be unknown. Of course, *ME* may learn that $(not\ p)$ holds for some predication p even if p is not subject to the CWA. In such a case, it stores that negative literal in its world model. Therefore, in checking whether a literal p is true or false, where p is not subject to the CWA, *ME* will not conclude "unknown" merely because p is absent from its world model – it also checks whether the negation is explicitly known. *ME* updates its knowledge of the non-occluded properties of an entity only when it (re)visits its location, and its knowledge of the occluded properties of that entity when it takes appropriate actions to (re)discover them.

In the case of autoepistemic predications, *ME* judges their truth value by an introspection algorithm rather than closed-world inference. In evaluating a predication of form $(knows\ SUBJ\ (that\ p))$ (e.g., $(knows\ ME\ (that\ (can\_talk\ guru)))$), the algorithm considers the two cases $SUBJ = ME$ and $SUBJ \neq ME$. In the first case, *ME* uses the methods in the previous paragraph to determine whether p is true, false or unknown, and judges the autoepistemic predication to be true, false or false respectively (i.e., in the latter two cases, *ME* does not know p to be true). In the case $SUBJ \neq ME$, *ME* judges the epistemic predication true only if *SUBJ* is identical to the subject of p (thus making a similar knowledge assumption about other agents as for itself (Kaplan and Schubert, 2000)), and false otherwise (a negative closure assumption that could be weakened). The method for predications of form $(knows\ SUBJ\ (whether\ p))$ is much the same. But in the case $SUBJ = ME$, when p 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 p, and false otherwise.

ME's inference capabilities are important in its attempt to confirm or disconfirm action preconditions, including knowledge preconditions in actions like asking an agent whether p is true (viz., not knowing whether p is true, but believing that the agent knows whether p is true). Similarly, the inference capabilities are crucial in answering questions, including ones that test *ME*'s introspective capabilities. Given a question, *ME* will either inform the user if it doesn't know the answer, or otherwise verbalize its answer(s) as English sentence(s).

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 $(has\_IQ\ moe\ 50)$.

5 EXPERIMENTAL AGENT

We implemented a version of *ME* in our framework to demonstrate self-motivated question-answering, and the benefits of *ME*'s opportunistic behavior resulting from its continual, deliberate, reward-seeking planning.

5.1 Simulated World

We situate *ME* in a very simple simulated world, yet one sufficiently subtle to illustrate the above features. This world is inhabited by animate agents *ME* and *guru*, along with inanimate objects *pizza*, *juice*, *note* and *piano*. There are four locations *home*, *grove*, *plaza* and *company*, with roads *path*1, *path*2 and *path*3 as shown in Figure 1. Object *pizza* is edible and at *home*; *juice* is potable and at *plaza*; *note* is readable, at *home*, and contains the knowledge whether *piano* is playable; *piano* is playable and at *home*. For demonstration purposes, *pizza* and *juice* are presumed inexhaustible. Agent *guru* knows whether *pizza* is edible, whether *note* is readable, and whether *juice* is potable.

Initially *ME* is at home, 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 objects at *home*, *ME* knows whether *pizza* is edible, that *juice* is at *plaza*, and that *guru* located at *grove* can talk and knows whether *juice* is potable. *ME* has the following operators at its disposal (where associated reward levels are indicated

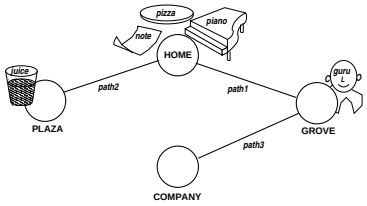


Figure 1: The experimental world.

parenthetically): *eat* (2 * *ME*'s hunger level at the onset of the action), *drink* (50), *walk* (3 - *ME*'s fatigue level at the onset of the action), *sleep* (*ME*'s fatigue level at the onset of the action), *play* (3), *read* (5), *ask other agents whether something is true* (3), and *answer the user's yes/no and wh-questions* (10).

Any state *s* reached by *ME* in its projective lookahead has an inherent value, based on the following criteria comparing *s* with its parent state. Answering a user question yields 50 utility points; becoming not bored gains *ME* 1 point; becoming more tired (thirstier, hungrier) costs *ME* points equal to the increase in *ME*'s fatigue (thirst, hunger) level. Conversely, leaving a user question unanswered costs *ME* 20 points; becoming bored costs *ME* 1 point; becoming less tired (thirsty, hungry) gains *ME* points equal to the decrease in *ME*'s fatigue (thirst, hunger) level. These criteria, along with the operators' anticipated rewards, suggest that *ME* likes drinking and answering user's questions the most.

A potentially multi-step action faces the possibility of interference by two types of exogenous events, namely rain and fire, but only fire may disrupt the action (e.g., sleeping, traveling). Spontaneous rain has a 33% chance of starting; once begun, it has a 25% chance of stopping. As long as there is no rain, a spontaneous fire has a 5% chance of starting; once started, it has a 50% chance of dying by itself, and also goes out when there is rain.

5.2 Question-Answering

ME's dialogue interaction is handled uniformly via *ME*'s planning and procedural attachment capabilities. Input questions currently must be expressed in the same syntax as that for symbolically representing knowledge in the simulated world. A yes/no question is prefixed with *ask-yn*; a wh-question, with *ask-wh*. Though more than one question can be entered at a time, *ME* handles questions independently, and it may be some time until *ME* chooses to answer a question as the next best action to perform.

The following example is self-explanatory.

```
>> (listen!)
((ask-yn user (is_tasty pizza))
 (ask-yn user (not (is_bored ME))))

ACTION: (ANSWER_YNQ (IS_TASTY PIZZA))
Answer: (ME DOES NOT KNOW WHETHER PIZZA IS TASTY)

ACTION: (ANSWER_YNQ (NOT (IS_BORED ME)))
Answer: (IT IS NOT THE CASE THAT ME IS BORED)
```

A wh-question of the form (*ask-wh user r*) must have at least one variable (indicated by prefix ?) in *r*. For instance, (*not* (*likes ME* ?*x*)) corresponds to the question "what does *ME* not like?" while (*is_at* ?*y* ?*z*) translates to "where is every entity located?". In computing the answer(s), *ME* attempts to unify *r* with facts in its current knowledge base; for each set *s* of bindings found for the variables in *r*, *ME* forms the corresponding answer by replacing variables in *r* with bindings in *s*. *ME* uses the weakened CWA to verbalize its responses as follows.

```
>> (listen!)
((ask-wh user (is_tired_to_degree ME ?x))
 (ask-wh user (is_animate ?y))
 (ask-wh user (is_bored ?z)))

ACTION: (ANSWER_WHQ (IS_TIRED_TO_DEGREE ME ?X))
ANSWER: (ME IS TIRED TO DEGREE 1.5)

ACTION: (ANSWER_WHQ (IS_ANIMATE ?Y))
ANSWER: (ME IS ANIMATE) (GURU IS ANIMATE)

ACTION: (ANSWER_WHQ (IS_BORED ?Z))
ANSWER: (ME DOES NOT KNOW WHETHER ANYTHING IS BORED)
```

6 RESULTS

We give further examples of *ME*'s question-answering, and empirical results showing the advantages of *ME*'s opportunistic behavior (due to its self-motivated, deliberate, continual planning) in the context of the simulated world in Section 5.1.

6.1 Extended Example of QA

The following, including knows-whether and knows-that questions, is a concatenation of several dialogue exchanges between *ME* and the user, showing only *ME*'s actions to answer user questions and *ME*'s corresponding verbalized English answers while omitting system output. The examples showcase *ME*'s ability to introspect positively and negatively using the relaxed CWA.

```
ACTION: (ANSWER_YNQ (KNOWS GURU
          (WHETHER (LIKES GURU PIZZA))))
ANSWER: (GURU KNOWS WHETHER GURU LIKES PIZZA)
ACTION: (ANSWER_YNQ (KNOWS GURU
          (WHETHER (LIKES ME PIZZA))))
ANSWER: (ME DOES NOT KNOW WHETHER GURU KNOWS
          WHETHER ME LIKES PIZZA)
ACTION: (ANSWER_YNQ (CAN_FLY GURU))
ANSWER: (IT IS NOT THE CASE THAT GURU CAN FLY)
ACTION: (ANSWER_YNQ (KNOWS GURU
                      (THAT (CAN_FLY GURU))))
ANSWER: (GURU KNOWS THAT IT IS NOT THE CASE
          THAT GURU CAN FLY)
```

6.2 Opportunistic Behavior

We report experiments wherein *ME*'s behavior shows both opportunism and foresight. The success of a run is measured in terms of the *net utility* (NU), summed over the entire sequence of actions and corresponding states. We conducted ten independent runs of *ME*'s opportunistic behavior, using a five-level lookahead with a branching factor of 3 for state nodes at each level. Each run was for 20 steps where at each step *ME* chose *and* performed a/the next best action. *ME* obtained an NU of 181.30 averaged over the ten runs ranging from NU 101.0 to NU 217.75.

The following shows *ME*'s sequence of 20 actions for one of the runs, wherein no spontaneous fire occurred, and *ME*'s NU was 191.0. Each action was logged along with its first iteration, anticipated duration, and time in the simulated world when that iteration began; also, each multi-step action was logged again with its final iteration.

```
((EAT 4.0 PIZZA HOME) 1 1 0)
((WALK HOME PLAZA PATH2 0.0) 1 2 2)
((WALK HOME PLAZA PATH2 0.0) 2 2 3)
((WALK PLAZA HOME PATH2 1.0) 1 2 5)
((WALK PLAZA HOME PATH2 1.0) 2 2 6)
((SLEEP 2.0 0.0) 1 8.0 8)
((SLEEP 2.0 0.0) 4 8.0 11)
((WALK HOME GROVE PATH1 0.0) 1 2 13)
((WALK HOME GROVE PATH1 0.0) 2 2 14)
((ASK+WHETHER GURU (POTABLE JUICE) GROVE) 1 1 16)
((WALK GROVE HOME PATH1 1.0) 1 2 18)
((WALK GROVE HOME PATH1 1.0) 2 2 19)
((SLEEP 2.0 1.0) 1 8.0 21)
((SLEEP 2.0 1.0) 4 8.0 24)
((EAT 2.0 PIZZA HOME) 1 1 26)
((WALK HOME PLAZA PATH2 0.0) 1 2 28)
((WALK HOME PLAZA PATH2 0.0) 2 2 29)
((DRINK 2.0 JUICE PLAZA) 1 1 31)
((WALK PLAZA HOME PATH2 1.0) 1 2 33)
((WALK PLAZA HOME PATH2 1.0) 2 2 34)
((SLEEP 2.0 0.0) 1 8.0 36)
((SLEEP 2.0 0.0) 4 8.0 39)
((WALK HOME GROVE PATH1 0.0) 1 2 41)
((WALK HOME GROVE PATH1 0.0) 2 2 42)
((ASK+WHETHER GURU (READABLE NOTE) GROVE) 1 1 44)
((WALK GROVE HOME PATH1 1.0) 1 2 46)
((WALK GROVE HOME PATH1 1.0) 2 2 47)
((SLEEP 2.0 1.0) 1 8.0 49)
((SLEEP 2.0 1.0) 4 8.0 52)
((EAT 2.0 PIZZA HOME) 1 1 54)
((READ 0.0 NOTE) 1 1 56)
((WALK HOME PLAZA PATH2 0.0) 1 2 58)
((WALK HOME PLAZA PATH2 0.0) 2 2 59)
```

ME's chosen seemingly best action in itself alone may not be immediately rewarding to *ME*, but rather is anticipated by *ME* to lead to a most rewarding sequence of actions. For example, though *ME* might not get a reward for walking from *home* to *grove*, it may very well foresee a high reward resulting from traveling to *grove* to meet *guru*, asking *guru* to learn about *note*'s readability (knowledge which *ME* does not have), going *home* where *note* is, and eventually reading *note*. Thus, the use of the reasoned, projective lookahead enables *ME* to exhibit foresight and opportunism in its behavior.

6.3 Goal-Directed Behavior

We made *ME* purely goal-directed by modifying its metrics of rewards and penalties for actions and states in the lookahead calculation. We designated drinking *juice* as *ME*'s sole goal and made *ME* uninterested in actions other than asking *guru* to acquire beverage knowledge, traveling to reach *guru* and *juice* and drinking *juice*. This was done by setting the anticipated rewards of operators other than *drink* and *ask+whether* to 0, leaving the anticipated reward of the former unchanged at 50 and that of the latter at 3, rewarding the acquisition of knowledge and the relieving of thirst in a state reached, while ignoring other changes in a state reached.

ME's sequence of actions was deterministic as follows, yielding an actual NU of 61.0. This was manually calculated using the *ME*'s original (just as in the opportunistic) metrics of rewards and penalties for actions and states reached. To find out about the potability of *juice*, *ME* must walk from *home* to *grove* and ask *guru*; with the projective lookahead, having knowledge about *juice* will incline *ME* to walk to *plaza* and drink *juice* there.

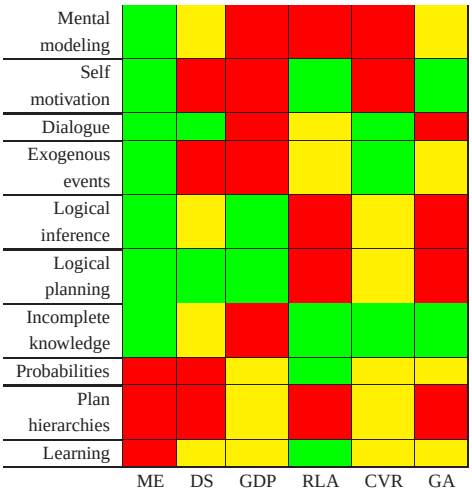
```
((WALK HOME GROVE PATH1 0) 1 2 0)
((WALK HOME GROVE PATH1 0) 2 2 1)
((ASK+WHETHER GURU (POTABLE JUICE) GROVE) 1 1 3)
((WALK GROVE HOME PATH1 1.0) 1 2 5)
((WALK GROVE HOME PATH1 1.0) 2 2 6)
((WALK HOME PLAZA PATH2 2.0) 1 3 8)
((WALK HOME PLAZA PATH2 2.0) 2 3 9)
((WALK HOME PLAZA PATH2 2.0) 3 3 10)
((DRINK 4 JUICE PLAZA) 1 1 12)
```

Compared with the opportunistic behavior of Section 6.2, the NU of 61.0 here is substantially lower. Instead of doggedly pursuing the sole goal of drinking *juice*, *ME* saw and seized *additional* opportunities in Section 6.2, including eating *pizza* to sate hunger, asking *guru* to find out *note*'s being readable, sleeping to relieve fatigue, and reading *note* for enjoyment. The results establish that *ME* benefits from awareness and exploitation of opportunities for immediate rewards, not only distant ones.

7 CONCLUSIONS

Our framework has demonstrated the feasibility of combining planning, inference and dialogue in a completely self-motivated cognitive agent. *ME* as exemplified empirically 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

Table 1: Domain-independent Capabilities [column - ME agent, DS: dialog systems, GDP: goal-directed planners, RLA: RL agents, CVR: cognitive (virtual) robots, GA: game agents].



general knowledge about itself and its environment, including both introspective (autoepistemic) knowledge and knowledge about mental states of other agents.

Table 1 summarizes the distinctive features of our framework vis-à-vis five previously studied frameworks.[2] A cell is green to indicate that the surveyed systems of that kind all exhibit that capability, red to indicate that none of them have that capability, or yellow to indicate that some of them have (some of) that capability.

In our future work, we expect to allow for degrees of uncertainty in *ME*'s knowledge. As well, we plan to implement learning by modification of the agent's anticipated utility for certain actions and states, so as to favor rewarding action sequences and avoid adverse sequences based on the agent's past experience. Lastly, we expect to generalize the framework to do hierarchical planning, which is essential for dealing with the complexities of dialogue about the real world.

REFERENCES

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

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

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

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

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

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

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

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

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

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

Nilsson, N. J. (1984). Shakey the robot. Technical Report 323, AI Center, SRI International.

Sacerdoti, E. D. (1975). A structure for plans and behavior. Technical Report 109, AI Center, SRI International.

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

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

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

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

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

[2] Sample references are (Ferguson and Allen, 1998; Allen et al., 2007) for DS, (Fikes et al., 1972; Kautz and Selman, 1992; Sacerdoti, 1975) for GDP, (Singh et al., 2002; Walker, 2000; Singh et al., 1994) for RLA, (Vere and Bickmore, 1990; Nilsson, 1984; Tacke et al., 2004; Ferrein et al., 2004) for CVR, and (Dinerstein et al., 2008; Davies and Mehdi, 2006) for GA.