Appears in AAAI'02 workshop on Cognitive Robotics (CogRob'02).

Adventure Games: A Challenge for Cognitive Robotics

Eyal Amir
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720-1776
eyal@cs.berkeley.edu

Patrick Doyle
Computer Science Department
Stanford University
Stanford, CA 94305-9020
pdoyle@cs.stanford.edu

Abstract

This paper presents a set of challenges for cognitive robotics in the context of a text-based adventure game. Games in this class have many challenging properties for cognitive robotics, including incompletely specified goals, an environment revealed only through exploration, actions whose preconditions and effects are not known *a priori*, and the need of commonsense knowledge for determining what actions are likely to be available or effective. These qualities require an agent that is able to use commonsense knowledge, make assumptions about unavailable knowledge, revise its beliefs, and learn what actions are appropriate. At the same time, more traditional robotics problems arise, including sensing, object classification, focusing on relevant features on a situation, reasoning within context, and decision-making, all within a large state space. In this paper we introduce the game and its environment, elaborate upon the properties of both as they pertain to cognitive robotics, and argue that this is a highly advantageous venue for exploring cognitive robotics issues.

Introduction

Researchers in cognitive robotics are interested in building control systems for robots (physical or virtual) that use high-level commonsense knowledge, yet are able to act and react in a complex environment. Work in this field varies from the development of high-level control languages (Levesque *et al.* 1997; Boutilier, Reiter, & Price 2001; Andre & Russell 2000), to controlling robots using implemented reasoners based on formal languages (Baral & Tran 1998; Shanahan 1998; 2000; Amir & Maynard-Reid II 1999; 2001), planners (Finzi, Pirri, & Reiter 2000; Bacchus & Kabanza 2000; Doherty & Kvarnström 2001) and the formalization of reasoning about action and events (Shanahan 1997; Gustafsson & Doherty 1996; Giunchiglia, Kartha, & Lifschitz 1997; McCain & Turner 1997; Bacchus, Halpern, & Levesque 1999; Reiter 2001).

This paper uses a concrete problem to examine a set of core issues facing cognitive robotics. These issues include

- Specifying the goal/utility/value/reward,
- Acting in a domain with incomplete knowledge of the possible situations, objects in the world, actions available, their preconditions, or their effects,
- Avoiding uselessly repetitive behaviors,
- Coping with domains that require representing large state spaces and reasoning with them,
- Reasoning in situations that require commonsense knowledge (either learned or provided),
- Accumulating knowledge about the environment and available actions, and integrating it with existing knowledge,
- Revising knowledge efficiently and in a semantically correct way,
- Embodying solutions to these problems in a sound and semantically clear agent architecture.

As (Laird & van Lent 2001) have argued, computer games provide a natural and challenging platform for artificial intelligence research. In order for an agent to perform effectively in a computer game, it must provide all of the major components of an intelligent system, namely intelligent behavior, robustness, real-time responsivity, commonsense reasoning, planning, learning, and communication abilities.

In this paper we present a particular game, a text-based adventure called *CRAG* (Cognitive Robotics Adventure Game) built within a MUD environment (*Multi-User Dungeon* or *Dimension*; for more on MUDs see (Curtis 1992; Curtis & Nichols 1994; Carroll *et al.* 2001)). In its task, it is similar to many other adventure games such as *Zork* (Blank & Lebling 1980) and *Rogue* (Mauldin *et al.* 1984). The agent must complete a game whose ultimate goal is incompletely specified at the outset, and must act in an environment whose properties are only gradually revealed as it advances through the game. Exploration, experimentation, and timely, flexible reasoning are emphasized over protracted ratiocination relying upon access to all relevant knowledge.

In the next section we will describe the game and its underlying environment in more detail to make clear why it is a suitable testbed for cognitive robotics research. We then look at how this list of problems might be addressed in our game, and where the challenges lie.

Properties of an Adventure Game

A traditional adventure game allows a single player to explore a fantastic environment filled with opponents and treasures. Progress is made by exploring and manipulating the

environment in order to defeat enemies and solve complex puzzles. Often, but not always, the player has a general idea of an ultimate task to be completed (such as defeating the evil wizard, saving the space station from destruction, finding the cure for a terrible disease, etc.). For our purposes, the salient properties of a traditional adventure game environment are[1]:

1. The environment is relatively static. The player is the principal actor and the majority of effects are the results of his actions. If the player drops an object and departs, the object can be expected to remain until the player returns.
2. Effects of actions are local. Effects are also ordinarily also immediate, so a player pushing a button may expect to perceive the consequences directly.
3. The environment is discrete. The environment is composed of objects (caves, coins, swords, trolls, levers, doors, etc.) distinct from one another.
4. Actions are discrete. A player may turn a knob, open a door, climb a staircase, light a lantern, but actions do not overlap in time and are atomic.
5. There is no noise. Unless the designer explicitly introduces it, there is no uncertainty either in sensing or in action. Items do not slip from the player's grasp, the player does not have difficulty moving through the door, hearing what another character says, etc.
6. Communication is symbolic. Both perception and action occur through a symbolic text-based interface, so low-level sensory issues don't have to be dealt with. While they form an interesting line of research, these issues are not in the core of cognitive robotics. This property allows us to place an agent within a complex and dynamic environment without also forcing us to deal with all the sensory issues involved in the real world.
7. The player does not initially have enough information to solve the game. The player does not know the disposition of objects or their uses, the proper solutions to puzzles, the ways to defeat opponents, etc. These can only be learned through observation and experimentation.
8. From any given state, it is always possible to solve the game. Some early adventure games permitted players to act in such a way as to make the game unsolvable (throwing the sword into a pit made the dragon unkillable) but current philosophy holds that the game should prevent the player from taking such actions. Thus there are no actions that should never be attempted.
9. Commonsense knowledge is required. In most cases it is not made explicit what action will achieve an effect, and the player must guess at likely actions. That bottles contain liquids, a light source is needed to see in a dark area, and even that a dragon's breath can melt ice are facts the game assumes the player will know.

[1]We may relax those or add to them in any particular implementation, as detailed in the description that comes after the following list.

While we have implemented a traditional adventure game, we have chosen to do it in an environment that permits much more sophisticated qualities (Doyle 2002). MUDs were inspired by, and are superficially similar to, text-based adventure game worlds, but have several other critical qualities:

1. MUDs are programmable. The MUD platform[2] we use contains its own object-oriented variant of C, giving us the flexibility of a general programming language, but one that is tailored to the creation of these environments.
2. MUDs are server-based, persistent worlds that allow multiple simultaneous connections. Thus we can experiment with a single agent or with collections of agents operating independently, cooperatively, etc.
3. We can simplify any of the above game conditions. Thus we can assist an agent by providing additional information in circumstances where commonsense knowledge would ordinarily be required; we can introduce explicit goals to guide its behavior; we can make the environment entirely static except for the direct results of an agent's behavior, and so forth.
4. We can relax any of the above constraints, making the problem more challenging. Thus we can make the environment very dynamic, with objects and properties regularly changing; we can add noise, so that actions do not always succeed or perception is not entirely reliable; we can have actions with temporally- or spatially-distant effects or with nondeterministic effects; we can model action in continuous spaces, if we are concerned about subtle variations in the agent's actions.
5. Elements of the environment are running code rather than database objects; they can perform arbitrary computations autonomously or when the agent interacts with them. For example, when an agent examines a door, the door may provide different sensory data depending on its state, the agent's properties, the environment's properties, etc.

The Problems Presented by Adventure Games

We believe that the adventure game setting is well-suited to experimental work in cognitive robotics, as it introduces all of the problems a general cognitive robotic architecture would need to deal with, while allowing us considerable control over the environment. Here we consider our list of core issues in the context of such an environment.

Goal, Reward, Value, Utility

The goal in CRAG is to find a mountain fortress where a fabulous treasure is rumored to be hidden. The game begins at the home of an eccentric magician who may know how to reach the fortress. Initially, the agent has no knowledge of the environment beyond basic topological relationships, no knowledge of available actions other basic movement and item manipulation (such as `go north` or `get lantern`) that we would regard as commonsense actions, and no goal

[2]There are many MUD servers freely available; we are using the MudOS platform with the Lima MUD libraries.

other than the ultimate goal of finding the treasure. Exploration and experimentation are necessary to gain the knowledge needed to form subgoals, determine the proper actions, and complete the game.

Problem: Goal specification The first problem is how to specify a goal in an adventure game. Traditionally (e.g., (Reiter 2001)), goals are specified as conditions that specify a set of *goal situations*[3]. However, at any point prior to completing the game, the agent is not aware of all the actions, fluents, and states in the space. As a result, traditional planning techniques do not apply; until the game is solved, no set of known actions will provably lead to a goal state. The existence of such a set is a traditional assumption (e.g. (Lifschitz 1986; Boddy & Dean 1989; Pednault 1989; Latombe 1991; Levesque 1996; Blum & Furst 1997; Finzi, Pirri, & Reiter 2000)).

If we wish to use a goal to guide the actions of the agent, we must allow more sophisticated ways of specifying goals or using them. We can also allow the agent to form more goals to drive its inference and actions (e.g., (Laird, Newell, & Rosenbloom 1987)). There are several scenarios in which goals may be applied.

In the first scenario, the agent possesses both a goal and all of the knowledge about the environment and its actions required to produce a plan. In this case the agent may execute the plan and achieve its goal with certainty. This is the simplest and least likely case.

Secondly, the agent may have some goal when it encounters a situation that resembles a situation in which it achieved a comparable goal in the past. In this case the agent may form a plan by adapting its previous solution to the new circumstance. Of course, the plan may be rendered invalid by changes in the environment or by the dissimilarities between the situations, but this may be a useful approach when the situations are closely related, such as when an agent has learned to fetch a repair manual from the library to repair a broken machine and then encounters a second machine. A similar case is one in which we choose to find a plan that will *possibly* succeed, but not requiring certainty.

A third scenario arises when an agent's commonsense leads it to believe it can achieve a goal through some actions that *should* exist, although it does not yet know that all these actions actually do exist. For example, an agent encountering a fearsome dragon, and having commonsense knowledge that *dragons are evil creatures* and *evil creatures exist to be defeated* (at least in adventure games), will form a goal to defeat the dragon. It does not know what action will defeat the dragon, but it determines that there should be such an action, and forms its goal accordingly.

The agent can also use a goal to form subgoals. An agent exploring the environment may, upon seeing a closed door, form a goal to open the door by turning the knob. If the door is locked, the agent might use its commonsense knowledge to infer that there is a key somewhere to unlock the door, and add new goals to acquire any key it sees and try all keys it possesses in the lock. The agent cannot prove that it can accomplish these goals; they are in some sense opportunistic and depend upon the results of its exploration and experiments. However, it can use *default rules* (e.g., (McCarthy 1980; Reiter 1980)) to record such hypotheses, absorb such information from the environment or hold such background knowledge.

Lastly, the agent can form *information goals*, in which it seeks to visit any unexplored area, examine any unfamiliar object, observe the effects of trying an unfamiliar action, and so forth. In an environment in which the agent knows very little, these kinds of goals lead the agent to the knowledge it needs in order to solve the other goals listed above.

As in the SOAR paradigm (Laird, Newell, & Rosenbloom 1987), the agent may form many goals that are never satisfied, either because they are unnecessary (the agent may have a goal to try a key it has previously seen on the door, but discovers a second key that works first) or because they are impossible (because the agent's assumptions about the world are invalid; in the case of the door, perhaps there is no key and it must be opened in another way). This requires that a goal-directed agent be able to entertain multiple simultaneous goals and be able to switch focus among them, rather than considering how to solve a single goal to the exclusion of any others it might consider.

One advantage of the game world is that the agent can assume the game is intended to be solvable; therefore, any lack of knowledge about the environment that prevents it from acting to solve the game can be corrected. Moreover, at any time there are one or more actions or action sequences that the agent can perform to increase its knowledge or to satisfy one of its other goals.

Problem: Reward/utility specification and learning The notion of *reward functions* appears in several contexts in AI related to problem solving. In the context of Markov Decision Processes (MDPs) (Boutilier, Reiter, & Price 2001), the reward $R(s, a)$ refers to the instantaneous reward that the agent receives by performing action a in state s. The *utility* of choosing an action in a situation, $U(a, s)$ typically reflects risk, choice, preference and likelihood (see e.g., (Chajewska 2002)). Formally, utility functions are directly derivable from state value functions or reward functions and vice versa. In the MDP and reinforcement learning literature this is the optimal Q function, given a reward function $R(s, a)$.

The problem of playing CRAG can be modeled as a delayed *reinforcement learning* problem (Kaelbling, Littman, & Moore 1996), though not as one in which the reinforcement is delayed until the game is complete, because any feedback after the game is solved will not be useful to the agent. Instead, the agent's progress through the game could be interpreted as a sequence of episodes (each corresponding, perhaps, to a puzzle in the game) and reinforcements provided at the end of every episode. However, this suffers the disadvantage of coupling the credit assignment problem with a paucity of data, and is unlikely to enable the agent to improve its performance in any game of reasonable length.

[3]Our discussion is not limited to *goal situations*, and it carries over to other types of goals, such as *maintenance goals*, *periodic goals*, and others.

An alternative may be to use the agent's commonsense knowledge about the domain to develop a reward function. For example, it may not be advantageous to perform a dangerous action such as opening a lion's cage. If a room is locked, then there is probably some interest in finding a way to open it, if there is nothing else that makes sense. Picking up an object and adding it to our belongings is many times advantageous. Hierarchical decomposition of the problem into abstract sub-problems may lead to more readily-available rewards that can be used as a substitute in the absence of an overall reward function.

The literature on utility functions distinguishes several classes of utility functions and utility descriptions of interest in our context. Additive utility functions, multiplicative utility functions and multilinear utility are all decomposable utility functions that are relatively easy to reason about and construct from simple utilities given to state features in a domain. Conditional independence and context-based independence play important roles in combining utility functions from smaller fragments (Bacchus & Grove 1995). Also, learning the utility function of a human could provide our agent with a suitable utility function for this game. Similar techniques are used in decomposing reward and value functions (e.g., (Guestrin, Koller, & Parr 2002)).

Problem: Value specification and learning The notion of a *value function* for states, $V(s)$, arises in the context of decision problems in which the value of a state is the sum of the reward the agent receives for being in this state and the cumulative reward that the agent will receive in the future if it follows an optimal policy. Instead of specifying a reward, as discussed above, we can specify a value for a state, or a heuristic function that approximates this value. For example, the value of a state may be composed of the number of items in the agent's belongings, the number of rooms already visited, or in some number of "significant" actions that have been performed[4]. Such a heuristic function could be derived from the agent's knowledge and be updated with new knowledge it accumulates over the course of the game.

A good value function seems to be more valuable than an accurate reward function in our game. The former is a direct influence on the actions that the agent needs to perform at any moment, while the latter can serve to define this value function, but not in any easy way.

Situations in the World

Problem: The state space is revealed gradually In CRAG, as in the real world, most knowledge is acquired through exploration. Prior to interacting with the environment, the agent does not know its structure (the locations, their relationships, the routes between locations) or its contents. There is no way to know where a corridor leads without walking to its far end; there is no way to know what is in a cupboard without opening it and looking inside.

[4]A distinction might easily be drawn between insignificant activities, such as ordinary movement, picking up and dropping objects, sensing, etc., and significant activities, such as the use of an object that alters the environment in some way.

The nature of the game, however, does permit the agent to make some general assumptions about its structure. The environment is discrete, so there is a fixed and finite number of distinct locations, and there is a well-defined network of connections among the locations. There is a standard (though not necessarily complete) set of actions for navigating among locations. Objects are likewise discrete and finite and there are standard operators for performing basic manipulations (such as picking them up, putting them down, examining them, etc.).

Actions Available in a Situation

Problem: Actions are revealed gradually The actions in CRAG vary with the situation the agent is in. We do not know what actions are available to us in a situation before we arrive at that situation. For example, before entering a room we do not know what objects are inside or what actions are available on those objects. In one room we may find a chest that can be opened, while in another room we might find a candle which we can light if we have matches.

The effect of this problem is that the agent cannot ordinarily produce a plan to achieve an arbitrary goal, since it may well be the case that achieving that goal requires actions unknown to the agent. Worse, the goal may not even be achievable. All the agent can know is that its present knowledge, properly applied, will ultimately enable it to perform any actions necessary to achieve the ultimate goal of completing the game, whether or not it already knows those actions.

Note that the agent's lack of knowledge about available actions also creates the problem of finding out what those actions are. An ordinary human in an adventure game literally guesses at actions based upon assumptions about the properties of the world. Seeing a large weed, and possessing a spray can of weed killer, a human player will generally think to `spray weed with weed killer` and will try syntactic variations on that theme until successful or until the game clearly indicates that the idea won't work. An agent must either have sufficient commonsense knowledge to make a similar conclusion ("sufficient" being "human-level" in the general case, although we suppose that an experimental agent might only be encoded with "pertinent" knowledge when attempting to solve any of a class of adventure games), or the agent must be informed in some way by the environment what actions are permissible, even if the environment does not clearly indicate what their preconditions and their effects are. A principled way of exploring this second approach is discussed in (Doyle 2002).

Action Preconditions, Direct Effects and Ramifications

Problem: Inferring and learning preconditions and effects of actions Besides not knowing what actions are available to us before getting to the situation in which we need to apply them, our agent may not be aware of the preconditions or effects of the available actions. Some of the effects are revealed to the agent upon acting, but the preconditions remain to be determined, and other effects may have occurred that the agent may not have perceived. For example, pushing a button found in an attic may turn off the light

(an effect we perceive), but it may also cut all electric power to the house. If the power was out before pushing the button it would instead turn the power back on.

Our agent is required to act before knowing all of the preconditions and effects of it actions. Since the goal is to solve the game, rather than to provide an explanation of how the game may be solved, we could say that it is not entirely necessary for the agent to understand the effects of its actions; merely by blundering around it might reach the goal. More realistically, since there may be important actions the agent can undo (pulling and pushing levers, turning dials back and forth), it is likely to need to learn to associate its actions with changes to the state of the world. For example, when it learns that the power is out for the entire house after pressing the button in the attic, if it assumes that the world is deterministic, it can conclude that this is an effect of pressing the button when the power is on for the house. It may also generalize this to preconditions and effects for pressing the button when the power is off. This last conclusion is required for the agent to try pressing the button in order to return the power back on.

Very Large State Space

Problem: Scaling Most AI knowledge representation, planning, and reasoning techniques scale poorly. CRAG currently includes over a hundred locations and dozens of objects, each with its own location, properties, and possibly internal state information. A simple enumeration of the state space would be exponentially large in the number of locations and objects. This search space is too large for brute-force exploration and trial-and-error. Even if it were fully known, planning or finding an optimal (or approximate) action policy is neither simple nor fast using traditional planning and MDP solution techniques. The additional lack of knowledge makes this problem even more intractable.

Need Commonsense, Learned or Inserted

Problem: Determining what commonsense knowledge is needed Many of these problems are simpler if different sorts of commonsense knowledge are available to the agent. Some of the simplest examples of such commonsense knowledge being put to use are (Levesque *et al.* 1997; Bacchus & Kabanza 2000; Boutilier, Reiter, & Price 2001). This type of commonsense knowledge restricts the search space of planners, but can be applied to other goal-driven or utility-drive reasoning tasks.

Different facts about general domains can be used or learned during the game. We know that doors can typically be opened and closed, that keys unlock them and that items can be picked up and dropped off. We can classify objects into an object-oriented hierarchy, registering common properties for doors, windows, chests, cups, tables, walls, etc.

There is also dramatic use for default assumptions here. We can assume that actions have only their observed effects, and that their effects are typically local. We may also assume that actions' preconditions are local. Typically, we can act upon an object only when it is present, and some actions' effects are reversible by performing those actions again. Such knowledge is not always correct, and the agent should be able to disregard it if needed, but without it there is little chance of inferring much about the world.

Problem: Deciding how to use commonsense knowledge If we classify objects into an object-oriented hierarchy, we can conclude that what we learn about one chest is typically true of other chests (similarly for doors, keys, buttons, etc). This allows us to learn the effect of classes of actions much faster than with no commonsense knowledge.

Inferring the effects of actions and their preconditions is also a problem where commonsense knowledge should play an important role. Some work on abduction (Ayeb, Marquis, & Rusinowitch 1993; Baral 2000), explanation (Minton *et al.* 1989; Poole 1989; McIlraith 1998), and diagnosis (Reiter 1987; de Kleer & Williams 1987; McIlraith 1997; Thielscher 1997; Darwiche 1998) are examples. There is also some work on inferring actions' properties (Schmill, Oates, & Cohen 2000), and on model-free reinforcement learning (Watkins 1989; Watkins & Dayan 1992) that are very relevant in this context.

Finally, using default information is crucial in deciding on the actions that are likely to be useful and avoiding actions that are harmful. The principle question of how to specify this knowledge in a consistent and useful manner is a research topics that have been addressed extensively by many researchers. However, the practical and technical questions that are associated with specifying such default knowledge in contexts similar to ours has received little attention (Poole 1989; Pinto & Reiter 1993; Nirkhe, Perlis, & Kraus 1993).

Memory and Accumulation of Knowledge

Problem: The organization of knowledge An application that needs to include commonsense knowledge in large quantities needs a disciplined way of organizing this knowledge. Some of the work on knowledge representation, including frame systems (Fikes, Farquhar, & Rice 1997; Koller & Pfeffer 1998) and object-oriented knowledge representations (Koller & Pfeffer 1997; Amir 1999) can be used for this. Learning into such object-oriented representations (e.g., (Friedman *et al.* 1999)) is required to maintain and develop this knowledge as the game proceeds. However, learning probabilistic object-oriented representations is not applicable as-is to our problem, and there has been no learning approach that can learn logical object-oriented knowledge (Inductive Logic Programming (ILP) (Lavrač & Džeroski 1994) focuses on other problems (however, see the work of (Gil 1994; Benson & Nilsson 1995)).

Several architectural approaches are also relevant. Approaches such as *blackboard* architectures (Nii 1986; Hayes-Roth 1985) may be useful in combining knowledge sources. The context-based CYC (Lenat 1995) and similar approaches can serve as a basis for the design of a smaller knowledge base that can be used for our agent. The reinforcement learning paradigm for agent design (e.g., (Sutton 1990)) can serve as a base upon which to build, but the technical problems involved in the straightforward application of reinforcement learning to our domain need to be resolved

(e.g., including commonsense knowledge in reinforcement learning agents is still an open problem, and the sparsity of the space (we rarely arrive at the same world state more than once) prevents the straightforward application of reinforcement learning techniques).

Problem: Learning and revision of knowledge For any architecture for knowledge organization and action we need to specify how knowledge is revised. In the logical AI literature this has been discussed extensively from philosophical points of view (e.g., (Gärdenfors 1992; Goldszmidt & Pearl 1992; Baral *et al.* 1992; Boutilier 1994; Lehmann 1995)) and some practical methods (e.g., (Eiter & Gottlob 1992; Val 1992; Liberatore & Schaerf 1997)) and applications (and revised philosophical assumptions) have emerged (Williams 1997; 1998). In the probabilistic AI literature belief revision is related to statistical machine learning (Duda, Hart, & Stork 2001; Jordan 1998) and to inference given observations (Pearl 1988; Cowell *et al.* 1999). Approaches for ILP have worked on the similar problem of classifying concepts by learning Horn rules (Gabbay *et al.* 1992; Lavrač & Džeroski 1994; Bratko & Muggleton 1995).

Current approaches to learning and revising knowledge virtually disregard the problem of revising first-order knowledge. For example, there is little attention to the discovery of new objects, actions or relationships in the world. There is little work on learning the properties of actions and models of the world in partially observable domains. The principal reason may be that the problems involved are too difficult and that straightforward approaches do not scale well for reasoning in such domains and with such knowledge. Nonetheless, our CRAG game may pose some of those problems in ways that are easier to attack. Because it is a concrete problem whose complexity we can control, we can focus our attention on those features we can address with solutions that are both formally and practically reasonable.

Solution in a Semantically Sound Way

The problems listed above are not specific to the CRAG adventure game, nor are they new for the cognitive robotics community or the AI community at large. They are fundamental for any truly autonomous agent capable of acting in a complex and dynamic world. The advantage of CRAG is that it allows us to examine them in a controlled environment in which we can easily change the problems to be solvable, and then gradually increase the difficulty step by step. We can also transcend some of the sensory elicitation problems that are typically encountered in physical agents (robots).

The challenge in this game is threefold. First, to build an agent that can play it with a performance that is close to that of humans. Second, to construct a formal theory (be it logical or probabilistic) that accounts for the way an autonomous agent should play this game, including the knowledge involved, the assumptions made and the architecture used. The final challenge is to combine a successful agent and a satisfactory theory into a single framework.

Conclusions

This paper shows that through examining a specific adventure game we can isolate many problems that have not been addressed or have been only partially addressed in the cognitive robotics literature and the AI literature in general. We described the problems that we observed in this game and outlined possible approaches for dealing with several. Examining those problems within the structure of our adventure game system allows us to move from addressing them in an *ad hoc* manner and toward discovering *well-founded* approaches for their solution.

We are in the process of implementing the logical interface and a simple agent that will play CRAG in a reactive way. We have already examined several architectures that may be suitable for a more *intelligent* agent, including GOLOG, reinforcement learning, subsumption architectures, blackboard architectures, and others. These analyses are omitted here for lack of space, but they are available in an extended version of this manuscript (Amir 2002). We are currently extending two of those architectures to begin to address the problems we have described in this paper.

Acknowledgements

The first author wishes to thank Stuart Russell, Greg Lawrence, David Andre, Kamin Whitehouse, Barbara Engelhardt and the other members of the AI-Agents reading group in UC Berkeley, and also Mike Genesereth, Aarati Parmar, Scott Rixner and other members of the FRG/MUGS groups for discussions on related topics.

References

Amir, E., and Maynard-Reid II, P. 1999. Logic-based subsumption architecture. In *Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99)*, 147–152.

Amir, E., and Maynard-Reid II, P. 2001. LiSA: A robot driven by logical subsumption. In working notes of the CommonSense'01 symposium.

Amir, E. 1999. Object-Oriented First-Order Logic. *Electronic Transactions on Artificial Intelligence* 3, Section C:63–84. http://www.ep.liu.se/ej/etai/1999/008/.

Amir, E. 2002. Adventure games: A challenge for cognitive robotics (full version). Available at the author's website (http://www.cs.berkeley.edu/~eyal/papers).

Andre, D., and Russell, S. J. 2000. Programmable reinforcement learning agents. In *Proceedings of the 13th Conference on Neural Information Processing Systems (NIPS'00)*, 1019–1025. MIT Press.

Ayeb, B. E.; Marquis, P.; and Rusinowitch, M. 1993. Preferring diagnoses by abduction. *IEEE Transactions on SMC* 23(3):792–808.

Bacchus, F., and Grove, A. 1995. Graphical models for preference and utility. In *Proc. UAI '95*, 3–10. MK.

Bacchus, F., and Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence* 116(1-2):123–191.

Bacchus, F.; Halpern, J. Y.; and Levesque, H. J. 1999. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence* 111(1–2):171–208.

Baral, C., and Tran, S. C. 1998. Relating theories of actions and reactive control. *Electronic Transactions on Artificial Intelligence* 2(3-4):211–271.

Baral, C.; Kraus, S.; Minker, J.; and Subrahmanian, V. S. 1992. Combining knowledge bases consisting of first-order theories. *Computational Intelligence* 8(1):45–71.

Baral, C. 2000. Abductive reasoning through filtering. *Artificial Intelligence* 120(1):1–28.

Benson, S., and Nilsson, N. 1995. Reacting, planning, and learning in an autonomous agent. *Machine Intelligence* 14:29–64.

Blank, M., and Lebling, D. 1980. Zork. Infocom.

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1–2):279–298.

Boddy, M., and Dean, T. 1989. Solving time-dependent planning problems. *Proc. International Joint Conference on Artificial Intelligence (IJCAI)* 2:979–984.

Boutilier, C.; Reiter, R.; and Price, B. 2001. Symbolic dynamic programming for first-order MDPs. In *IJCAI '01*, 690–697. MK.

Boutilier, C. 1994. Unifying default reasoning and belief revision in a modal framework. *Artificial Intelligence* 68:33–85.

Bratko, I., and Muggleton, S. 1995. Applications of inductive logic programming. *Communications of the ACM* 38(11):65–70.

Carroll, J. M.; Rosson, M. B.; Isenhour, P.; Ganoe, C.; Dunlap, D.; Fogarty, J.; Schafer, W.; and Metre, C. V. 2001. Designing out town: MOOsburg. *International Journal of Human-Computer Studies* 54:725–751. Available online at http://www.idealibrary.com.

Chajewska, U. 2002. *Acting Rationally with Incomplete Utility Information*. Ph.D. Dissertation, Stanford University.

Cowell, R. G.; Dawid, A. P.; Lauritzen, S. L.; and Spiegelhalter, D. J. 1999. *Probabilistic networks and expert systems*. Springer-Verlag.

Curtis, P., and Nichols, D. A. 1994. MUDs grow up: Social virtual reality in the real world. In *COMPCON*, 193–200.

Curtis, P. 1992. Mudding: Social phenomena in text-based virtual realities. In *Proceedings of the 1992 Conference on the Directions and Implications of Advanced Computing*.

Darwiche, A. 1998. Model-based diagnosis using structured system descriptions. *Journal of AI Research* 8:165–222.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32:97–130.

Doherty, P., and Kvarnström, J. 2001. Talplanner: A temporal logic based planner. *AI Magazine*. Accepted for publication.

Doyle, P. 2002. Believability through context: Using "knowledge in the world" to create intelligent characters. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems*. ACM Press. To appear.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2001. *Pattern classification*. Wiley, 2nd edition.

Eiter, T., and Gottlob, G. 1992. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence* 57(2-3):227–270.

Fikes, R.; Farquhar, A.; and Rice, J. 1997. Tools for assembling modular ontologies in ontolingua. In *Proc. AAAI '97*, 436–441. AAAI Press.

Finzi, A.; Pirri, F.; and Reiter, R. 2000. Open world planning in the situation calculus. In *Proc. AAAI '00*, 754–760.

Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *Proc. Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99)*, 1300–1307. MK.

Gabbay, D.; Gillies, D.; Hunter, A.; Muggleton, S.; Ng, Y.; and Richards, B. 1992. The rule-based systems project: Using confirmation theory and non-monotonic logics for incremental learning. In Muggleton, S., ed., *Inductive Logic Programming*. Academic Press. 213–230.

Gärdenfors, P., ed. 1992. *Belief Revision*. New York: Cambridge University Press.

Gil, Y. 1994. Learning by experimentation: Incremental refinement of incomplete planning domains. In *Proceedings of the 11th International Conference on Machine Learning (ICML-94)*, 10–13.

Giunchiglia, E.; Kartha, G. N.; and Lifschitz, V. 1997. Representing Action: Indeterminacy and Ramifications. *Artificial Intelligence*. to appear.

Goldszmidt, M., and Pearl, J. 1992. Rank-based systems: A simple approach to belief revision, belief update, and reasoning about evidence and actions. In Nebel, Bernhard; Rich, Charles; Swartout, W., ed., *Proc. KR '92*, 661–672. Cambridge, MA: Morgan Kaufmann.

Guestrin, C.; Koller, D.; and Parr, R. 2002. Multiagent planning with factored mdps. In *Proceedings of 14th NIPS*. MIT Press.

Gustafsson, J., and Doherty, P. 1996. Embracing occlusion in specifying the indirect effects of actions. In *Proc. 5th Intl Conf. on Knowledge Representation and Reasoning (KR '96)*, 87–98.

Hayes-Roth, B. 1985. A blackboard architecture for control. *Artificial Intelligence* 26:251–321.

Jordan, M., ed. 1998. *Learning in graphical models*. MIT Press.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: a survey. *Journal of AI Research* 4:237–285.

Koller, D., and Pfeffer, A. 1997. Object-oriented Bayesian networks. In *Proc. of the 13th Conf. on Uncertainty in Artificial Intelligence (UAI-97)*, 302–313. Morgan Kaufmann.

Koller, D., and Pfeffer, A. 1998. Probabilistic frame-based systems. In *Proc. AAAI '98*, 580–587.

Laird, J., and van Lent, M. 2001. Human-level ai's killer application: Interactive computer games. *AI Magazine* 22(2):15–26.

Laird, J.; Newell, A.; and Rosenbloom, P. 1987. SOAR: An architecture for general intelligence. *Artificial Intelligence* 33(1):1–64.

Latombe, J.-C. 1991. *Robot Motion Planning*. Kluwer Academic Publishers.

Lavrač, N., and Džeroski, S. 1994. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.

Lehmann, D. 1995. Belief revision, revised. In *IJCAI-95*, 1534–1540.

Lenat, D. B. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM* 38(11):33–38.

Levesque, H.; Reiter, R.; Lesprance, Y.; Lin, F.; and Scherl, R. 1997. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming* 31:59–84.

Levesque, H. 1996. What is planning in the presence of sensing. In *Proc. AAAI '96*, 1139–1146.

Liberatore, P., and Schaerf, M. 1997. Reducing belief revision to circumscription (and vice versa). *Artificial Intelligence* 93(1-2):261–296.

Lifschitz, V. 1986. On the semantics of STRIPS. In Georgeff, M. P., and Lansky, A., eds., *Reasoning About Actions and Plans*. Los Altos, California: MK. 1–9.

Mauldin, M. L.; Jacobson, G. J.; Appel, A. W.; and Hamey, L. G. C. 1984. Rogomatic: A belligerent expert system. In *Proc. Fifth Nat. Conf. Canadian Soc. for Computational Studies of Intelligence*.

McCain, N., and Turner, H. 1997. Causal theories of action and change. In *Proc. AAAI '97*.

McCarthy, J. 1980. Circumscription—A Form of Non-Monotonic Reasoning. *Artificial Intelligence* 13:27–39.

McIlraith, S. 1997. Representing action and state constraints in model-based diagnosis. In Senator, T., and Buchanan, B., eds., *Proc. AAAI '97*, 43–49. Menlo Park, California: American Association for Artificial Intelligence.

McIlraith, S. 1998. Explanatory diagnosis: Conjecturing actions to explain observations. In Cohn, A. G.; Schubert, L.; and Shapiro, S. C., eds., *Proc. KR '98*. San Francisco, California: MK. 167–177.

Minton, S.; Carbonell, J. G.; Knoblock, C. A.; Kuokka, D. R.; Etzioni, O.; and Gil, Y. 1989. Explanation-based learning: A problem solving perspective. *Artificial Intelligence* 40:63–118.

Nii, P. 1986. Blackboard systems. *AI Magazine* 7(2).

Nirkhe, M.; Perlis, D.; and Kraus, S. 1993. Reasoning about change in a changing world. In *Proceedings of FLAIRS'93*.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann.

Pednault, E. P. D. 1989. ADL: exploring the middle ground between STRIPS and the situation calculus. In *Proc. KR '89*, 324–332.

Pinto, J., and Reiter, R. 1993. Temporal reasoning in logic programming: A case for the situation calculus. In *Proceedings of the Tenth International Conference on Logic Programming*, 203–221.

Poole, D. 1989. Explanation and prediction: An architecture for default and abductive reasoning. *Computational Intelligence* 5(2):97–110.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13 (1–2):81–132.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32:57–95.

Reiter, R. 2001. *Knowledge In Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press.

Schmill, M. D.; Oates, T.; and Cohen, P. R. 2000. Learning planning operators in real-world, partially observable environments. In *Proceedings of the 5th Int'l Conf. on AI Planning and Scheduling (AIPS'00)*, 246–253.

Shanahan, M. 1997. *Solving the Frame Problem, a mathematical investigation of the common sense law of inertia*. Cambridge, MA: MIT press.

Shanahan, M. 1998. A logical account of the common sense informatic situation for a mobile robot. *Electronic Transactions on Artificial Intelligence* 2:69–104.

Shanahan, M. 2000. Reinventing shakey. In Minker, J., ed., *Logic-Based Artificial Intelligence*. Kluwer.

Sutton, R. S. 1990. Integrated architecture for learning, planning and reacting based on approximating dynamic programming. In *th Int'l conference on machine learning (ICML'90)*, 216–224. MK.

Thielscher, M. 1997. A theory of dynamic diagnosis. *Electronic Transactions on Artificial Intelligence* 1(4):73–104.

Val, A. D. 1992. Computing knowledge base updates. In Nebel, B.; Rich, C.; and Swartout, W., eds., *Proc. KR '92*. San Mateo, California: Morgan Kaufmann. 740–750.

Watkins, C. J. C. H., and Dayan, P. 1992. Q-learning. *Machine Learning Journal* 8(3/4). Special Issue on Reinforcement Learning.

Watkins, C. J. C. H. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation, King's College, Oxford. (To be reprinted by MIT Press.).

Williams, M.-A. 1997. Anytime belief revision. In *Proc. Fifteenth International Joint Conference on Artificial Intelligence (IJCAI '97)*, 74–80. MK.

Williams, M.-A. 1998. Applications of belief revision. In *Transactions and Change in Logic Databases*, 287–316.