Guiding Inference through Relational Reinforcement Learning

Nima Asgharbeygi, Negin Nejati, Pat Langley, and Sachiyo Arai

Computational Learning Laboratory
Center for the Study of Language and Information
Stanford University, Stanford CA 94305, USA
{nimaa, negin}@stanford.edu
langley@csli.stanford.edu
arai@tu.chiba-u.ac.jp

Abstract. Reasoning plays a central role in intelligent systems that operate in complex situations that involve time constraints. In this paper, we present the Adaptive Logic Interpreter, a reasoning system that acquires a controlled inference strategy adapted to the scenario at hand, using a variation on relational reinforcement learning. Employing this inference mechanism in a reactive agent architecture lets the agent focus its reasoning on the most rewarding parts of its knowledge base and hence perform better under time and computational resource constraints. We present experiments that demonstrate the benefits of this approach to reasoning in reactive agents, then discuss related work and directions for future research.

1 Introduction

A fundamental goal of artificial intelligence is to develop systems that demonstrate intelligent behavior in complex environments. Such systems should be capable of assessing situations, reasoning about them, and making informed decisions even when confronted with constraints involving time and computational resources. For example, an embodied agent can benefit greatly by drawing inferences (internal beliefs) about its immediate situation (as perceived through sensors) using knowledge about the world (inference rules).

Because we are concerned with reactive agents, we focus here on data-driven bottom-up approaches to inference, rather than query-based top-down ones. Such agents need a belief state about the world in order to make a decision and take an action in any situation, so bottom-up inference over relational inference rules is the natural choice. However, it is clear that a reactive agent operating under time constraints cannot afford to exhaustively make all possible inferences. Rather, like humans, it must give priority to drawing more important conclusions and delay others. Such an informed agent may overlook important items on occasion, but it can still respond rapidly and its performance will degrade gracefully as complexity increases. This approach differs significantly from most

AI research on efficient matching and inference, which has combined exhaustive methods with clever indexing schemes (Doyle [1], Forgy [2]).

In this paper, we assume that a knowledge-rich reactive agent cannot afford to make all possible inferences, and thus must focus its attention. We are interested in an "anytime" inference mechanism that achieves high utility by inferring the most useful inferences within a given time limit. Moreover, under no time constraint, it should generate the same belief state as the exhaustive inference mechanism. As an implication, we prefer not to modify the structure of inference rules (unlike previous work on speedup learning, such as Zelle and Mooney [3]), but rather to have an adaptive reasoning system that learns *over* the relational structure in order to use it more efficiently.

Our solution, called the Adaptive Logic Interpreter (ADLIN), consists of two components—a value-driven inference process that iteratively selects the implied instantiated inference rule with the highest expected utility, and a learning mechanism that estimates these utilities based on received rewards. The latter uses a variation of relational reinforcement learning over the logical structure of inference rules. It incorporates a generalization mechanism that models the values estimated for instances of each first-order inference rule using regression methods. This model is then used to estimate the initial expected utility for new instances of the corresponding inference rule.

We should mention that, since the problem involves relationally represented states and actions, it is naturally posed as relational reinforcement learning (Tadepalli *et al.* [4]). Nevertheless, our generalization mechanism differs from the methods employed in earlier works on this topic. For example, Dzeroski *et al.* [5] applied inductive logic programming methods to induce first-order regression trees as generalizers. However, our approach represents the generalization knowledge as a set of linear regression models over the first-order predicates. We claim that our approach to relational reinforcement learning uses the prior knowledge encoded in the relational structure of a given domain effectively, and that it is capable of generalizing across distinct objects of the same class and transferring to tasks of different sizes.

Furthermore, in contrast with traditional reinforcement learning over physical actions, our formulation deals with actions that are internal to the agent. More specifically, we interpret each inference step as a mental action taken at some internal state that updates the agent's belief state but not the physical world. We hypothesize that the resulting adaptive attention method will let reactive agents make informed inferences under time constraints, and thus respond appropriately in complex environments.

We begin by reviewing ICARUS, a reactive agent architecture that currently relies on exhaustive inference to characterize situations and decide on its actions. After this, we describe ADLIN's method for giving priority to high-utility beliefs and an associated mechanism for learning their values. Next, we formalize our hypotheses about the benefits of this method and report experimental studies that demonstrate them empirically. In closing, we discuss related research on controlled inference and suggest directions for future work.

2 Architectural Framework

Our vehicle for studying controlled inference has been ICARUS, a reactive architecture for physical agents that has been described at length in Choi *et al.* [6]. Here we summarize the framework briefly, emphasizing those aspects most relevant to our current topic. We believe our approach will be applicable to other knowledge-rich reactive agent architectures that include an inference component.

An ICARUS agent lives in an *environment* composed of a dynamic collection of objects whose attributes and mutual relations change over time. Like other agent architectures, ICARUS operates in cycles. On each iteration, descriptions of objects perceivable to the agent (*perceptions*) are deposited into a perceptual buffer from which the system bases its inferences and generates a belief state. The interpreter then finds which *skills* match against the resulting belief state, selects the best applicable skill instance, and executes it in the environment.

Skills are Prolog-like rules that let the agent respond to different situations in the environment. These are organized in a hierarchy, so that each skill calls on lower-level skills or executable actions. Each skill specifies initiation conditions that match against descriptions of perceived objects or inferred relations among those objects. Unlike many reactive frameworks, ICARUS bases its decisions not only on primitive perceptions but also on its inferred beliefs.

In this paper we are mainly concerned with ICARUS' inference mechanism which, on each cycle, generates the agent's belief state based on the perceptions and the domain knowledge. Knowledge about the domain is stored in a long-term conceptual memory as a set of *concept* definitions. Concepts are first-order logical inference rules, each stated in terms of relations that must hold among objects, relations that must not hold for them, and arithmetic tests. In addition, associated with each concept is a reward function that specifies its utility to the agent when an instance of the concept holds. Like skills, concepts are defined in terms of perceptual entities and lower-level concepts, thus producing a hierarchical structure.

Table 1 presents three concept definitions from the blocks world. The first concept determines if the perceived object `?b` is a block. The second concept defines a `left-of` relation between two blocks. It holds between two perceived objects `?b1` and `?b2` of type `block` (specified in `:percepts`) whenever `is-block` predicate holds for both of them and their x positions satisfy the condition specified in `:tests`, which simply states that the x position of `?b1` must be less than that of `?b2`. The reward associated with this concept is always zero. The relation defined by the next concept, `between`, can be interpreted similarly, but notice that its reward is a function of the attributes of the perceived objects, in this example the x positions of the blocks involved.

On each execution cycle, the architecture initiates its inference procedure by examining the lowest-level concepts at the bottom of the hierarchy and inferring matched instances. Inferring a concept instance means checking whether its conditions hold based on the current belief state and, if so, adding the instance to the belief state. ICARUS then proceeds up the hierarchy, checking concepts that include newly inferred elements in their definitions. The process recurses

Table 1. Three examples of ICARUS concept definitions from the blocks world.

```
(is-block (?b)
 :percepts  ((block ?b xpos ?x))
 :reward     0.0)

(left-of (?b1 ?b2)
  :percepts ((block ?b1 xpos ?x1)
             (block ?b2 xpos ?x2))
 :positives ((is-block ?b1)
             (is-block ?b2))
 :tests     ((< ?x1 ?x2))
 :reward     0.0)
```

```
(between (?b1 ?b2 ?b3)
 :percepts  ((block ?b1 xpos ?x1)
             (block ?b2 xpos ?x2)
             (block ?b3 xpos ?x3))
 :positives ((left-of ?b1 ?b2)
             (left-of ?b2 ?b3))
 :reward (* 10 (- 30 ?x3 ?x2 ?x1)))
```

upwards, continuing until the entire hierarchy has been processed or, in the time-constrained case, until reaching the deadline.

We will refer to this inference method as *exhaustive*, because it considers concepts in a bottom-up, breadth-first manner with no control over the reasoning strategy. Clearly, this approach will not scale well to complex domains in which the knowledge base is large. Even worse, when the agent operates under time constraints, early termination can produce an inaccurate description of the environment, which in turn can produce undesirable behavior for the agent.

3 A Method for Controlled Inference

In order to overcome the drawbacks of exhaustive inference, a reasoning system requires some way to focus its cognitive attention on useful candidates. In this section, we introduce ADLIN, our adaptive logic interpreter. We begin by presenting a method for *value-driven* inference that gives priority to beliefs with higher expected utilities. After this, we describe a learning method that estimates these utilities from experienced rewards. Finally, we consider ADLIN's generalization mechanism for compactly modelling the knowledge learned over belief instances. The resulting system should fare better than an exhaustive version when the number of possible inferences exceeds the number that can be made in the time the agent has available.

3.1 Value-Driven Inference

Our approach to value-driven inference assumes that, for each candidate belief, the agent computes an expected utility, which it then uses to select the next instance to consider. The technique incorporates an agenda mechanism that inserts items into a list sorted by their priority levels, selects the topmost item to process, and iterates. This method is flexible enough to control inference, yet simple enough to incur little computational overhead.

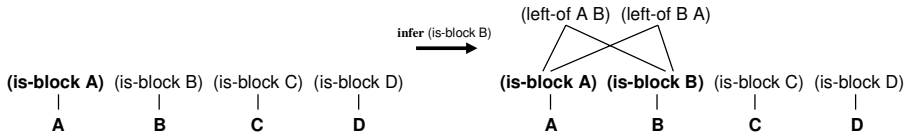

Fig. 1. A simple example of state and fringe update from the blocks world. The literals in bold belong to the belief state, whereas the others belong to the fringe.

More formally, we can specify a set of real values $V : \mathcal{U} \mapsto \mathcal{R}$ that is defined over $\mathcal{U}$, the set of all concept instances. These values differ from the rewards associated with concept definitions and may be specified in order to achieve a desired objective or reasoning strategy. As we will see later, ADLIN learns the values V to capture not only the immediate rewards, but also the future benefits of inferring different belief instances. Also notice that assigning scalar values to concept definitions would not be sufficient. The number of instantiated beliefs derived from one concept definition can be very large, in which case the inference system should be able to prioritize between them and infer only those with highest expected utility. This will let the system guide the inference process in a best-first manner, rather than a depth-first or breadth-first one.

Before describing the inference process, we should introduce a few more technical terms. We define the mental state, s_t, as the set of instances inferred to be true after t inference steps within the current execution cycle. Each inference step consists of selecting an instance that has not yet been inferred and checking whether it holds. This involves examining whether its child instances are believed and whether all its variable constraints and arithmetic tests are satisfied. At the beginning of each reasoning cycle, s_0 is empty. The *fringe* F_{s_t} at any state s_t is the set of all *inferrable* concept instances, that is, those instances not yet inferred within the current execution cycle whose children are already included in the state s_t.

A *valid action* a_u is an inference step that infers the instance $u \in F_{s_{t-1}}$ at the current step t. Finally, a *value-driven inference mechanism* is one that performs a valid action $a(V, F_{s_{t-1}})$ at each step t. In other words, every inference step depends entirely on the value assignment V and the contents of the current fringe. In this paper, we use a special *greedy* case of this mechanism class that always selects the highest value instance in the fringe. More precisely, we define $a_{greedy}(V, F_{s_{t-1}})$ as the action to infer the instance $\arg\max_{u \in F_{s_{t-1}}} V(u)$. This choice will prove reasonable when we consider the objective of the inference process shortly.

An example from the blocks world should illustrate the approach more clearly. Suppose the knowledge base consists of the three concept definitions introduced in Table 1 and there are four blocks in the environment. Figure 1 shows how executing an action updates the belief state and the fringe. Once the system selects inferring `(is-block B)` as the action, it checks whether this concept instance holds. Because `B` is a block, ADLIN adds this instance to its belief

state and, because `(is-block A)` is already in the state, `(left-of B A)` and `(left-of A B)` become inferrable, so it adds them both to the fringe.

3.2 Value-Learning Mechanism

As mentioned earlier, the nature and computation of expected utilities are contingent upon the objective of the value-driven inference mechanism. Here we define the objective in terms of reward, r_u, produced by an inference action a_u, which we define as

$$r_u = \begin{cases} R_u(x_u), \text{ if } u \text{ is inferred to true;} \\ 0, \qquad \text{otherwise,} \end{cases} \tag{1}$$

where R_u is the reward function associated with concept u and x_u denotes the attribute vector for perceived objects on which u depends. As we have noted, this reward function quantifies the desirability to the agent of instances for the given concept. When an instance does not match, it contributes zero reward. Therefore, r_u provides a reasonable measure for the immediate success of the corresponding inference action.

We assume the agent aims to maximize the cumulative absolute reward over each execution cycle, that is $W_T = \sum_{u \in s_t} |r_u|$, subject to the constraint that the t inference steps in this cycle take no more than T units of time. This objective is equivalent to minimizing $W_\infty - W_T$, which represents the error in calculated utility of the time-constrained system with respect to the utility calculated under no time constraints. In the terminology of value-driven inference, this objective translates to finding V values such that the inference strategy (approximately) maximizes W_T under a given time constraint T.

These values should capture not only the immediate expected reward for each inference action, but also its benefit for later inference steps. To this end, we adapt an approximate reinforcement learning method based on the account of states and actions presented earlier. We will let $Q(s_t, a)$ indicate the expected value of taking inference action a at state s_t. Recall that, unlike most work on reinforcement learning, our states and actions are completely internal to the agent. Furthermore, as equation (1) suggests, the source of reward is distributed over the entire knowledge base. This lets the learning element consider only the relevant parts of reward at each step.

Because the state space $\mathcal{S}$ of all possible states s_t can be intractably large, the classical tabular representations of $Q(s_t, a)$ for Q learning or $V(s_t)$ for value learning are impractical. Inspired by the MAXQ framework [7], we introduce a value decomposition that expresses the Q function in terms of V values:

$$Q(s_t, a) = \sum_{u \in s_t} V(u) + V(u_a) \ , \tag{2}$$

where u_a denotes the concept instance inferred by action a. While making the problem tractable, this approximation establishes a relationship between the

reinforcement learning method and value-driven inference. In fact, equation (2) lets us rewrite the standard stochastic Q function update rule for V values:

$$V(u) := \overline{\alpha}V(u) + \alpha[r_u + \max_{u' \in F^u_{s_t}} V(u')] \ , \tag{3}$$

where $F^u_{s_t} \subseteq F_{s_t}$ is the set of instances in the current fringe for which u is a child. Furthermore, we define α as

$$\alpha = \frac{1}{1 + visits(u)} \ , \tag{4}$$

in which $visits(u)$ indicates the number of updates performed on $V(u)$, and $\overline{\alpha}$ is given by

$$\overline{\alpha} = \begin{cases} 1, & \text{if } u \in s_t; \\ 1 - \alpha, & \text{if } u \notin s_t \ . \end{cases} \tag{5}$$

We derive the update rule provided in equation (3) from the standard stochastic Q function update rule [8]:

$$Q(s,a) := (1-\alpha)Q(s,a) + \alpha[R(s,a) + \gamma \max_{a'} Q(s',a')].$$

We can safely assume $\gamma = 1$, since we are dealing with a finite-horizon problem. Substituting the value decomposition in (2) and the definition of reward in (1) into this update rule gives

$$\begin{aligned} \sum_{u \in s_{t-1}} V(u) + V(u_a) &:= (1-\alpha)(\sum_{u \in s_{t-1}} V(u) + V(u_a)) \\ &\quad + \alpha[r_{u_a} + \max_{\text{valid } a'} (\sum_{u \in s_t} V(u) + V(u_{a'}))] \\ &:= (1-\alpha) \sum_{u \in s_{t-1}} V(u) + (1-\alpha)V(u_a) \\ &\quad + \alpha[r_{u_a} + \sum_{u \in s_t} V(u) + \max_{u_{a'} \in F_{s_t}} V(u_{a'})] \end{aligned} \tag{6}$$

Observe that

$$\sum_{u \in s_t} V(u) = \sum_{u \in s_{t-1}} V(u) + V(u_a) \cdot \mathbf{1}(u_a \in s_t) \ , \tag{7}$$

in which

$$\mathbf{1}(u_a \in s_t) = \begin{cases} 1, \text{ if } u_a \in s_t; \\ 0, \text{ if } u_a \notin s_t \end{cases} \tag{8}$$

As a result, the update rule simplifies to

$$V(u_a) := \overline{\alpha}V(u_a) + \alpha[r_{u_a} + \max_{u_{a'} \in F_{s_t}} V(u_{a'})] \ , \tag{9}$$

with $\overline{\alpha}$ being defined by (5). Notice that the only difference between the update rules in (9) and (3) lies in the argument of max. In fact, we have restricted the

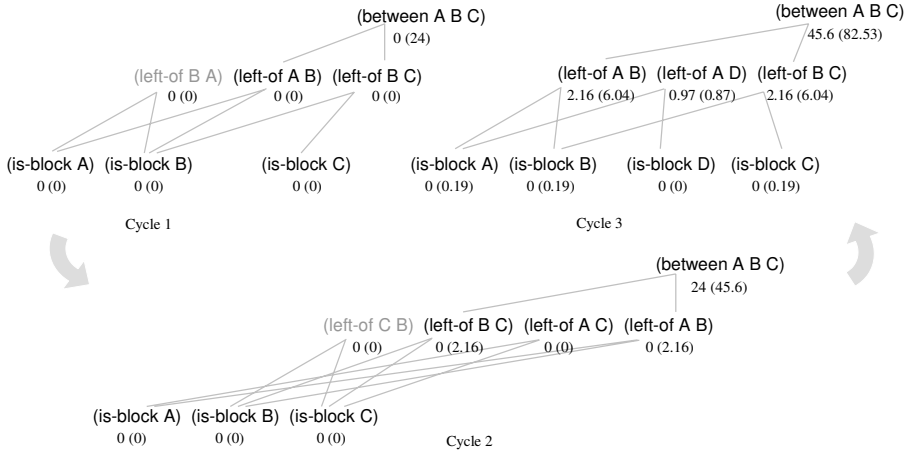

Fig. 2. An example of value propagation for three successive cycles in the blocks world. Updated values are shown in parentheses whereas others have been used to guide inference. The literals in grey have been inferred to be false.

argument in (3) to the set of instances that depend directly on u. This reflects the intuition that the values of instances which depend on u are more indicative of the desirability of inferring u than the values of other instances in the fringe.

Figure 2 shows how the values are learned and propagated in our example from the blocks world. Each diagram illustrates part of the agent's belief state at the end of the corresponding inference cycle. For an easier visualization of the procedure, the instances are ordered from left to right according to the order in which ADLIN infers them, the leftmost being the first. Having no previous experience in this domain, the system initializes the values to zero. As Table 1 shows, only the highest level concept, `between`, has nonzero reward and therefor the values remain zero until an instance of `between` is inferred as true. Once the system infers `(between A B C)`, it updates this belief's value in proportion to its reward. Equation 3 propagates this value to the immediate children, `(left-of B C)` and `(left-of A B)`, in the next cycle and consequently to the grandchildren—`(is-block A)`, `(is-block B)`, and `(is-block C)`—two cycles later. Note that the value updates occur after the instances are inferred and, as a result, the updated values participate in guiding the inference one cycle later.

3.3 Generalization Mechanism

The learning method described so far attempts to guide inference toward the most rewarding parts of the instance space $\mathcal{U}$ and maximize cumulative reward by propagating the rewards down the concept hierarchy. However, the value learned for a single instance lives only as long as the instance remains in $\mathcal{U}$. Thus, the agent would also benefit from a *generalization* mechanism that summarizes the learned knowledge about concept instances and stores it as compact models

for future use. ADLIN uses such models to initialize the expected value V for new candidate beliefs in the instance space. Without a generalization model, the inference system would need to relearn the value functions for every new instance, which is certainly undesirable.

Our generalization mechanism learns a linear model $h_c(x) = \theta^T x$ for every concept definition c. More precisely, it applies linear regression methods to incrementally update the linear model h_c associated with each concept definition c, using the training examples $S_c = \{(V(u), x(u)) \mid u \in \mathcal{U}_c \cap s_t\}$ at the end of each execution cycle. Here, $x(u)$ indicates the vector of numeric attributes for the perceptions that appear in concept instance u and $\mathcal{U}_c \subseteq \mathcal{U}$ denotes the set of all instances derived from concept definition c. Later, when a new concept instance is created, its corresponding linear model is evaluated to initialize the expected value V for the candidate belief. Given this prior knowledge, the inference mechanism can perform more efficiently, as we will demonstrate shortly. Despite their simplicity, these linear models appear to help significantly in improving ADLIN's overall performance.[1]

Figure 2 illustrates how generalization affects the priorities of inferring various instances. Once the value for `(left-of A B)` becomes 2.16, the generalization module takes advantage of the relational structure of the state space to update a linear model for the concept definition `left-of`. ADLIN uses the numeric attributes of blocks `A` and `B`, namely their x positions, and the value 2.16 to update the model, which it revises further based on other instances. However, when entirely new instances of the `left-of` concept become inferrable, the system uses this model to initialize their values. For example when `(left-of A D)` becomes inferrable after adding `(is-block D)`, it uses the learned model and the x positions of `A` and `D` to generate the initial value 0.97. In this example, the resulting value is higher than the value of `(is-block C)` and therefor the system prefers to infer `(left-of A D)` first.

Table 2 summarizes the ADLIN inference system, including its method for value-driven inference, its mechanism for reinforcement learning, and its technique for generalizing over instances of relational concepts.

4 Experimental Evaluation

Our primary goal for designing a value-driven inference mechanism, as stated earlier, is to make time-limited reasoning more effective by focusing cognitive attention on relevant parts of the instance space. To collect evidence that our approach has the desired effects, we carried out experiments within ICARUS that compared ADLIN's value-driven inference with exhaustive reasoning and also ADLIN without its generalization mechanism, all under time constraint and var-

[1] Another application of the generalization models, especially in highly dynamic environments, lies in updating the V values when the sensory attributes of the instance change. The update frequency should be increased with the frequency and variance of the change in sensory values.

Table 2. An outline of ADLIN inference process for one execution cycle.

1. INITIALIZATION
   i. At the beginning of each cycle, start with an empty state $s_0 = \emptyset$ and let $t = 0$.
   ii. If a new perceived object is added in this cycle, update $\mathcal{U}$ by adding the new instances and initialize their V values by evaluating their h_c functions.
   iii. If a perceived object is deleted, update $\mathcal{U}$ by removing all instances that depend on it. Initialize the fringe F_{s_0} with all primitive concept instances.
2. VALUE-DRIVEN INFERENCE AND VALUE-LEARNING
   Repeat the following steps run until time runs out:
   i. Set $t \leftarrow t + 1$ and infer the instance $u = \arg\max_{u' \in F_{s_{t-1}}} V(u')$.
   ii. Let $s_t = s_{t-1}$ and $F_{s_t} = F_{s_{t-1}}$.
   iii. If u is true, add it to s_t, compute its reward r_u, and update the fringe F_{s_t} .
   iv. Perform the V-value update for u:

$$V(u) := \overline{\alpha} V(u) + \alpha[r_u + \max_{u' \in F^u_{s_t}} V(u')] \ ,$$

   where $\alpha = 1/[1 + visits(u)]$ and $\overline{\alpha}$ is given by (5).
3. GENERALIZATION
   For each concept definition c:
   i. Consider the function $h_c(x) = \theta^T x$.
   ii. For every sample point in $S_c = \{(V(u), x(u)) \mid u \in \mathcal{U}_c \cap s_t\}$, perform the update:

$$\theta \leftarrow \theta + \alpha(V(u) - h_c(x(u))) \cdot x(u) \ .$$

ious circumstances. We present experimental evidence in two different domains, which we describe in detail below.

Naturally, we chose *reward accuracy* as the performance measure for our comparisons. Reward accuracy refers to the ratio between the cumulative reward obtained by the inference system under time constraints on a particular execution cycle to the total reward that would be accumulated on the same cycle by making all possible inference under no time constraints. This gives a measure on how successful our inference system is in guiding inference toward the most rewarding parts of the current instance space. Using the notation introduced in the previous section, we can express accuracy on some specific cycle as $A = W_T/W_\infty$. Recall that W_T denotes the cumulative absolute reward obtained in one cycle under time limit T. Similarly, we can state the reward error as $E = (W_\infty - W_T)/W_\infty$.

4.1 Blocks-world Domain

Our first experiments used a simple blocks-world environment because it gave us systematic control over factors of interest. The only objects in the environment are blocks placed in line, each with a name and a position specified as its distance from a reference point, which we call the origin. We used the three relational

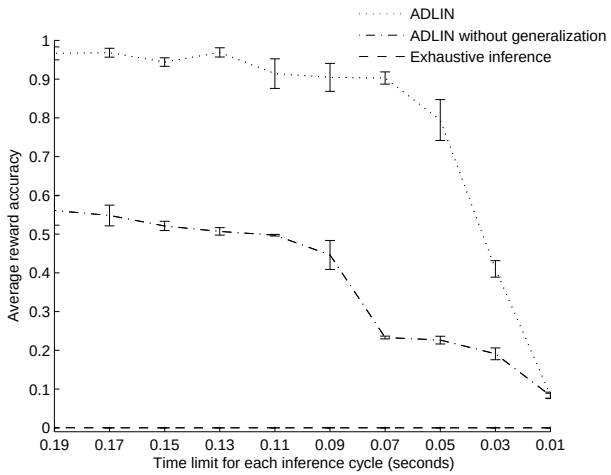

Fig. 3. Comparison between reward accuracies of three inference systems in the blocks-world environment over decreasing amounts of available time for each inference cycle.

concepts defined in Table 1. Clearly, when a large number of blocks are present, the number of feasible `between` relation instances will be enormous. We assume that the agent is located at the origin and prefers to interact with blocks sitting close to it. Therefore, we assigned a reward function to the highest level concept as a linear function that favors relations whose corresponding blocks are closer to the origin. The other two concept definitions had no assigned reward function.

We expected ADLIN to outperform exhaustive inference in terms of reward accuracy. However, we also anticipated that the degree of dominance would depend on factors such as the time limit for each inference cycle, domain complexity, and the rate of environmental change. First we considered the effect of time constraints on the behavior of three different inference systems: ADLIN, ADLIN without generalization, and exhaustive inference.[2] We expected that, as the available time for inference decreases, all three inference mechanisms would become less accurate, but we hypothesized that ADLIN's performance would degrade more gracefully.

We let each system learn for 200 cycles under a fixed time limit of 0.04 seconds in an initial world state with six blocks. We then tested the system under various time limits ranging from 0.19 to 0.01 seconds, in a dynamic environment in which a new block was added every 25 cycles, for a total of 100 cycles. At the end we measured the average inferred reward during the last 10 cycles and averaged the results over 20 independent runs. Figure 3 summarizes the results for the

[2] One might also consider an inference system that uses a greedy policy based on the immediate rewards of individual belief instances. In our example domains, however, there is no reward function assigned to low level concepts and hence such an inference mechanism should perform no better than exhaustive inference.

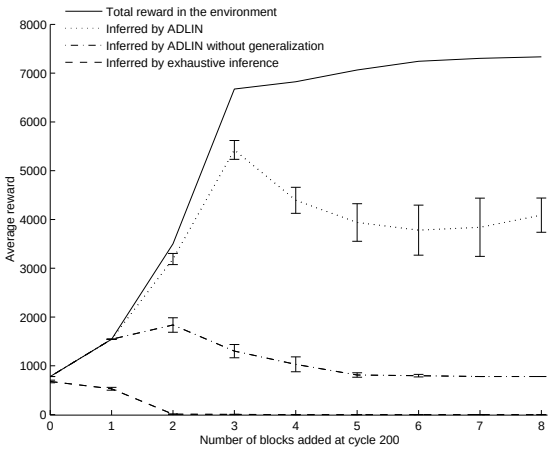

Fig. 4. Performance comparison between three inference systems over increasingly complex blocks-world environments.

three inference systems considered. The accuracy of the exhaustive inference method is almost zero for this range of time limits. Clearly, ADLIN provides a superior reward accuracy for most values of the time constraint. Its performance degrades only for extremely tight time limits (0.03 to 0.01 seconds), which is mainly caused by the computational overhead of its generalization mechanism.

Next we studied the effects of domain complexity, which refers to the size of the instance space. We let each system operate in a situation similar to the previous experiment, but we increased the complexity by abruptly adding objects, ranging from one to eight blocks, at cycle 200 and then ran the system for 100 more cycles. As before, we measured the average inferred reward over the final 10 cycles for 20 independent runs. Figure 4 depicts the results as a function of the number of blocks added. As expected, exhaustive inference performs very poorly. ADLIN without generalization gives better performance, but it cannot handle the excess complexity in the environment and, eventually, when the environment becomes too complex, it only obtains as much reward as it could in the initial world state.

In contrast, when equipped with generalization, ADLIN demonstrates substantially better performance, as shown by the dotted line in Figure 4. This result signifies that the generalization mechanism plays an important role in dealing with new inference instances. Notice, however, that ADLIN's performance degrades considerably when more than three blocks are added. This is because the first three new blocks are close to the origin and hence contribute a significant increase to the total reward, whereas the other new blocks are far from the origin and hence much less important. Nonetheless, these blocks distract ADLIN to some extent and cause the degradation in its performance.

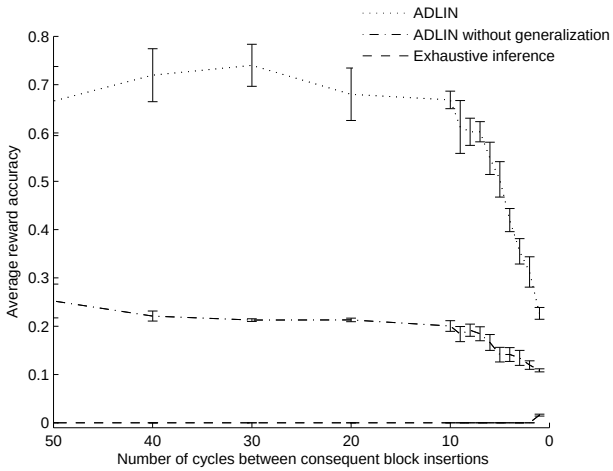

Fig. 5. Accuracy of different inference systems over increasing rates of environmental change in the blocks-world domain.

The last factor we considered was the rate of change in the environment. We set up a similar experiment to the one above but, after the 200-cycle learning period, instead of introducing an abrupt change, we inserted the new blocks one at a time with a specific number of cycles between consequent insertions. Thus, we systematically varied the rate of change in the world and measured the average performance of each inference system over its last 10 cycles. As Figure 5 illustrates, ADLIN is the most robust of the three systems across a wide range of rates. However, for very rapidly changing environments, even ADLIN's performance degrades significantly, as seen when we added one block per cycle. In the closing section, we propose some responses to this issue.

4.2 In-city Driving Domain

As a more realistic, and hence more interesting, domain for evaluating ADLIN we chose an in-city driving environment. This is an appropriate domain both because of its complexity and its inherent time constraints in collision-like situations. In this simulated environment [6], all objects take a rectangular form on a Euclidean plane. The simulator supports static objects such as road segments, intersections, lane lines, and buildings, as well as dynamic ones like vehicles. One of the vehicles is controlled by an ICARUS agent and all others follow realistic physical laws but with predetermined behavior.

The vehicle controlled by the agent can perceive objects that are in its field of view, defined by the radius of a circle centered at its current position. The objects are described by numeric attributes like distance, angle, relative velocity, and angular velocity. The ICARUS agent also perceives its own properties, including distance and angle with respect to lane lines, as well as its speed and the angle

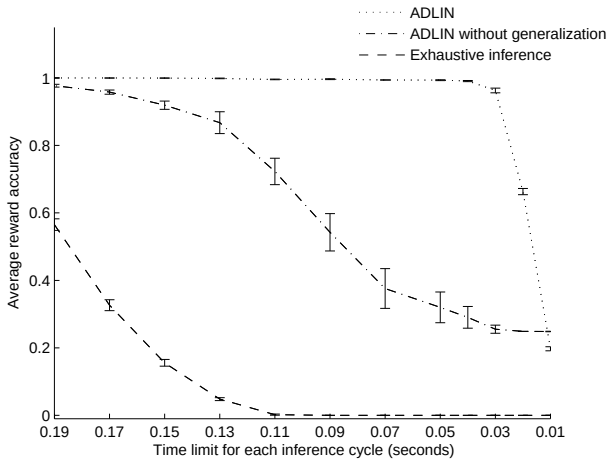

Fig. 6. Comparison between reward accuracies of three inference systems in the in-city driving environment over decreasing amounts of available time for each inference cycle.

of its steering wheel. The agent must drive the vehicle safely by staying on the right side of the road, making necessary turns, and avoiding collisions. These constraints produce a complex environment which requires reasoning about many objects with different priorities.

In our first experiment in the driving domain, we again considered time constraint as the independent variable, while holding complexity and rate of change fixed. Figure 6 summarizes the results for the three inference systems in our study, namely ADLIN, ADLIN without generalization, and exhaustive inference.

In this experiment, we had the agent drive around a block in the simulated city at a constant speed of 15 miles per hour. We let each system learn under a fixed time limit of 0.5 seconds while making a complete turn around the block. Then we measured the average reward inferred by the system when turning around the same block at the same speed, but under different time constraints ranging from 0.19 to 0.01 seconds. Figure 6 presents the resulting accuracies averaged over 20 independent runs for each system. ADLIN demonstrates a higher tolerance of time constraints, especially for time limits between 0.13 to 0.03 seconds. However, we observed a similar sudden degradation of performance for extremely tight time constraints as seen earlier in the blocks-world study.

In the driving domain, the rate of change and complexity of the environment are mainly determined by the agent's speed of driving. Therefore, we cannot vary these factors separately while keeping the other fixed. In response, our next experiment evaluated each inference system in the driving domain for different driving speeds. As the agent drives faster both complexity and the rate of change in the environment increase. The complexity increases since the overlap of the visible areas across the cycles decreases and therefore more of the perceived objects are new. Figure 7 shows the average reward achieved by different

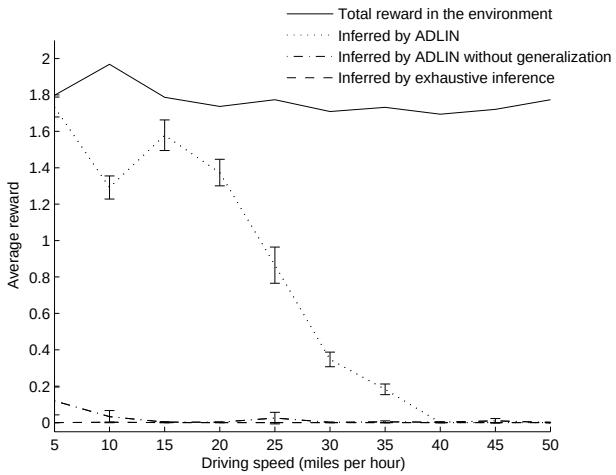

Fig. 7. Average inferred rewards achieved by different inference systems in the driving domain for different driving speeds.

inference systems for different driving speeds, compared to the total reward that is available in the environment.

We let each system learn under a time constraint of 0.4 seconds while the agent was driving on a straight street at a speed of 5 miles per hour. Then we tested the system under a 0.08-second time limit while driving at different constant speeds that ranged from 10 to 50 miles per hour. We computed inferred rewards over the entire test period, again averaged over 20 independent runs. The curves of Figure 7 show the superior performance of ADLIN for speeds under 35 miles per hour, although its reward accuracy decreases at higher speeds. Clearly, even ADLIN has difficulty in dealing with highly dynamic environments, here represented as driving speeds over 40 miles per hour. Again, we will return to this issue in Section 6.

5 Related Research on Inference and Learning

The challenge of reasoning under resource constraints is nearly as old as the field of artificial intelligence, but the problem has typically been neglected in classical theories of normative behavior. The common approach to mitigating the problem in practical intelligent systems has been to employ heuristic, and usually domain-dependent, control strategies to guide reasoning. For example, the meta-level reasoning system developed by Genesereth and Ginsberg [9] lets the designer write Prolog-like control clauses that specify how inference rules should be prioritized.

However, research on the topic of *bounded rationality* has attempted to deal with the problem in a domain-independent way. Early work by Simon [10] examined humans' reliance on satisficing strategies when confronted with complex

decision-making tasks. More recently, Horvitz [11] has discussed limitations of traditional normative approaches in dealing with real-world complexity and proposed adapting such methods to reason about the reasoning process itself. Similarly, Russell and Wefald [12, 13] sought to develop a theoretical framework for meta-reasoning that was based on probability and decision theory. Our approach shares the idea of considering computations as mental actions with Russell and Wefald analysis.

Other work has focused on learning control rules to reduce search or speed up processing (see for example Minton [14]), some of which has dealt with monotonic inference. Zelle and Mooney [3] combined explanation-based learning techniques with inductive logic programming ideas to learn such control conditions over inference rules. In a different approach, Cohen and Singer [15] used bootstrapping to learn similar rules. These approaches assumed a query-based, top-down inference mechanism and sought to modify the logic program itself in order to achieve performance gains. In our work, however, we consider a data-driven, bottom-up inference process that is more appropriate for reactive agents, for the reasons discussed earlier. In addition, our approach operates on top of a fixed logic program and modifies the way it is utilized by the logic interpreter.

Our approach has more in common with research on relational reinforcement learning, at least in its broad sense defined by Tadepalli *et al.* [4], and hierarchical reinforcement learning [7]. However, as mentioned earlier, our variation on relational reinforcement learning differs from the initial work by Dzeroski *et al.* [5] in its use of linear regression models distributed over first-order concepts to support generalization. Moreover, in contrast to most work on these topics, our states and actions are cognitive rather than physical and they are completely internal to the agent.

Our work also shares some of its basic ideas with the recent work by Guestrin *et al.* [16], including relational representation of states and actions, distributed reward over the relational structure of first-order predicates, additive approximation of value function, and class-based generalization. Nevertheless, there are significant differences in problem setting and approach. Guestrin *et al.* considered a planning problem modeled by a relational Markov decision process with fixed relations in the world, discrete attributes, and given transition probabilities. They also took a linear programming approach to solving their problem. In contrast, we have addressed an inference problem modeled as a reinforcement learning problem over a relational state structure with dynamic relations and continuous attributes. Our approach holds the promise of rapid learning and the ability to scale to large state spaces, which makes it closer in spirit to work on explanation-based reinforcement learning (e.g., Dietterich and Flann [17]).

6 Concluding Remarks

In this paper, we introduced the Adaptive Logic Interpreter (Adlin), which uses a value-driven inference mechanism combined with reinforcement learning to deal with the challenge of data-driven inference under time constraint. Our

experiments in dynamic domains showed that ADLIN performs well in guiding the reasoning process toward high-utility parts of the knowledge base and outperforms exhaustive inference system. This difference was especially large under short time limits and high rates of environmental change.

However, when the time constraint is extremely strict or the environment is changing too rapidly, ADLIN does not provide a satisfying performance gain. This is partly due to the computational overhead involved in the system and partly because, in the current system, every instance in the agent's belief state is inferred on the current execution cycle, regardless of whether it was true or false on the previous cycle. Although this simplification did not appear to be a crucial restriction in our experimental studies, it should certainly be addressed in future extensions.

One basic approach to dealing with this issue is to incorporate the simple idea behind truth maintenance systems and Rete matchers [2]. These systems maintain the truth value of each inference instance unless the evidence supporting that instance changes, in which case its truth value is updated. It should be straightforward to extend ADLIN to consider only instances with changed evidence as candidates for inference, from which it then selects high value instances to actually infer. This should let the system concentrate on the belief instances that need to be reinferred. Nevertheless, a more refined approach would consider the intensity of change in the environment and its corresponding effect on each belief instance. More precisely, we propose the idea of *probabilistically persistent belief*, in which the agent persists in maintaining a belief instance and only updates it with some probability that depends on its intrinsic variance given the amount of change in evidence for that instance. Such a probabilistic measure can be applied either to reduce the number of candidate inferences or to modulate their expected values to incorporate the effect of their change of evidence into the value-driven inference process.

Clearly, our design of ADLIN assumes that the domain's reward structure is additively decomposable over the concepts (first-order predicates). However, this does not impose any fundamental restriction on generality of our approach, for two reasons. First, interactions among different concepts in the reward function can be expressed by their relational structure. For example, imagine a predator/prey scenario in which proximity of a predator lowers the reward of proximity of a mate for a prey animal. This effect can be captured by defining a higher level concept that describes the relational situation (proximity of mate given the proximity of a predator) and assigning it a low reward function. Second, the behavior produced by a non-decomposable reward structure can often be well-approximated by a decomposable one. In the above example, suppose the reward function assigned to "predator proximity" (or to the concepts built on top of it) is higher than the reward associated with "mate proximity." Then ADLIN would pay more attention to the former and, should it hold, to reasoning on top of it, effectively being less concerned about the latter.

Finally, from the point of view of agent architectures, a complete attention mechanism must consider not only the concept instances that the agent believes,

but also the skills by which it interacts with its environment. Therefore, in the longer term, a promising direction is to extend our attention mechanism to cover skill inference and selection. Such an extension would require considering both the temporal effects and the top-down nature of skill execution. Despite the need for further improvements, ADLIN has already revealed its potential for the effective control of inference, which in turn has taken us closer to a practical attention mechanism for reactive agents.

References

1. Doyle, J.: A truth maintenance system. Artificial Intelligence **12** (1979) 231–272
2. Forgy, C.L.: Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence **19** (1982) 17–37
3. Zelle, J.M., Mooney, R.J.: Combining FOIL and EBG to speed-up logic programs. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Chambery, France, Morgan Kaufmann (1993) 1106–1111
4. Tadepalli, P., Givan, R., Driessens, K.: Relational reinforcement learning: An overview. In: Proceedings of the ICML-2004 workshop on Relational Reinforcement Learning, Banff, Canada (2004)
5. Dzeroski, S., Raedt, L.D., Driessens, K.: Relational reinforcement learning. Machine Learning **43** (2001) 7–52
6. Choi, D., Kaufman, M., Langley, P., Nejati, N., Shapiro, D.: An architecture for persistent reactive behavior. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems, New York, ACM Press (2004) 988–995
7. Dietterich, T.G.: Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research **13** (2000) 227–303
8. Mitchell, T.M.: Machine Learning. McGraw Hill, New York (1997)
9. Genesereth, M.R., Ginsberg, M.L.: Logic programming. Communications of the ACM **28** (1985) 933–941
10. Simon, H.A.: Administrative behavior. 2nd edn. Free Press, New York (1965)
11. Horvitz, E.: Reasoning about beliefs and actions under computational resource constraints. Journal on Uncertainty in Artificial Intelligence **3** (1989) 301–324
12. Russell, S., Wefald, E.: Principles of metareasoning. In: Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning, San Mateo, CA, Morgan Kaufmann (1989)
13. Russell, S., Wefald, E.H.: Do the Right Thing: Studies in Limited Rationality. MIT Press, Cambridge, MA (1991)
14. Minton, S.: Quantitative results concerning the utility of explanation-based learning. In: Proceedings of the Seventh National Conference on Artificial Intelligence, Saint Paul, MN, AAAI Press (1988) 564–569
15. Cohen, W.W., Singer, Y.: A simple, fast, and effective rule learner. In: Proceedings of the Forteenth National Conference on Artificial Intelligence. (1999) 335–342
16. Guestrin, C., Koller, D., Gearhart, C., Kanodia, N.: Generalizing plans to new environments in relational MDPs. In: Proceedings of International Joint Conference on Artificial Intelligence, Acapulco, Mexico (2003)
17. Dietterich, T.G., Flann, N.S.: Explanation-based learning and reinforcement learning: A unified view. Machine Learning **28** (1997) 169–210