Means-End Plan Recognition – Towards a Theory of Reactive Recognition

June, 1994

Technical Note 49

By:

Anand S. Rao
Australian Artificial Intelligence Institute

This research was supported by the Cooperative Research Centre for Intelligent Decision Systems under the Australian Government’s Cooperative Research Centres Program. This paper also appears in the Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning KRER-94, Bonn, Germany, May, 1994
Abstract

This paper draws its inspiration from current work in reactive planning to guide plan recognition using "plans as recipes". The plan recognition process guided by such a library of plans is called means-end plan recognition. An extension of dynamic logic, called dynamic agent logic, is introduced to provide a formal semantics for means-end plan recognition and its counterpart, means-end plan execution. The operational semantics, given by algorithms for means-end plan recognition, are then related to the provability of formulas in the dynamic agent logic. This establishes the relative soundness and completeness of the algorithms with respect to a given library of plans. Some of the restrictive assumptions underlying means-end plan recognition are then relaxed to provide a theory of reactive recognition that allows for changes in the external world during the recognition process. Reactive recognition, when embedded with the mental attitudes of belief, desire, and intention, leads to a powerful theory of integrated reactive planning and recognition. The primary contribution of this paper is in laying the foundations for such an integrated theory.
1 INTRODUCTION

Classical planning and plan recognition have received a great deal of attention within the Artificial Intelligence (AI) community. Classical planning deals with reaching a desired state of affairs from the current state by chaining through a given set of plan operators. Plan recognition, usually treated as the reverse process of planning, is concerned with inferring these operators based on observations.

Over the past decade the focus of research in planning has shifted from classical planning to reactive or situated planning [6]. Reactive planning is based on two premises: (a) the environment in which an agent is situated is continuously changing; and (b) agents situated in such environments have limited resources. This has led to the development of various architectures and techniques for guiding the agent in its decision-making process, for making agents commit to their decisions as late as possible and, once committed, to stay committed as long as possible, within rational bounds.

Research in reactive planning has led to the re-definition of the notion of plans. Plans are used in two different contexts: (a) plans as abstract structures or recipes for achieving certain states of the world; and (b) plans as complex mental attitudes interwoven in a complex web of relationships with the other mental attitudes of belief and desire [15]. Plans as recipes guide a resource-bounded agent in its decision-making process, thereby short-circuiting the time-consuming search through a possible space of solutions as done by classical planning. Plans as mental attitudes constrain the agent in its future decision-making by committing it to previously-made decisions. The latter are called intentions.

As noted by others\(^1\) this renaissance in planning, has had very little impact on plan recognition. A majority of the work within plan recognition [1; 9; 13] is still addressing the general problem of unconstrained plan recognition. Although some of these approaches use background knowledge (in terms of event hierarchies [9] and plans [13]) as heuristics to guide the general recognition problem, they have not attempted to use plans in the above sense to guide or constrain the recognition process of resource-bounded agents in a dynamic world.

The use of plans as recipes and as mental attitudes, to guide and constrain the recognition process, respectively, will be called reactive recognition.

Reactive recognition is applicable to a broad class of problems where agents have limited resources and the environment may change while the agents are doing their recognition. However, it makes two important assumptions: (a) the recognizing agent has complete knowledge of the plans of other agents that it is trying to recognize; and (b) under any given situation the recognizing agent has a small set of plans (i.e., hypotheses) that it is trying to recognize.

In this paper, we address a part of the reactive recognition process, namely the use plans as recipes to guide the recognition process and call it means-end plan recognition\(^2\). In addition to the two assumptions (a) and (b) of reactive recognition, means-end recognition also makes the following assumptions: (c) the occurrence of events in the external world is synchronous with the recognition of events by the agent, i.e., the agent cannot wait for an event to occur and there is no memory of all the past events that have occurred; and (d) the world is not changing while the agent is performing the recognition.

Due to its restrictive assumptions means-end plan recognition is of limited applicability. However, it lays the foundation on which to build a theory of reactive recognition. We present simple algorithms for means-end plan recognition that make use of plans, similar to those used in reactive planning systems, to determine what means must be observed in

\(^1\)Pollack [16] writes: "Yet, most research on plan recognition has taken place in isolation from the AI planning renaissance. Could a marriage of these two research projects bear any fruit?"

\(^2\)Analogously, we shall refer to the usage of plans to guide the planning process as means-end plan execution.
order to recognize certain ends. We introduce a dynamic agent logic that provides a logical semantics for means-end plan recognition and means-end plan execution. In this paper, we are predominantly concerned with the theoretical principles of means-end plan recognition and its relationship to the algorithmic operational semantics. While we do not envisage the use of means-end plan recognition in practical applications, we do see it as an important first step towards reactive plan recognition. Extensions of the means-end plan recognition algorithms to reactive recognition can be found elsewhere [19].

In spite of the initial skepticism towards reactive planning, the approach has been quite successful compared to classical planning. This is due to the fact that, in a substantial class of problem domains (such as road-traffic management [3], space shuttle diagnosis [8], air-traffic management [14], and air-combat modelling [18]) execution of actions and decision-making tasks can be analyzed and codified as plans, in a relatively simple manner. These domain-dependent plans can then be effectively used by an agent to react in dynamic domains under resource constraints. The need for reactive recognition was motivated by the fact that plan recognition in these domains is simpler than the more general problem. In particular, an agent in these domains is not attempting to recognize any arbitrary plan, but instead knows that it is attempting to recognize one out of a small set of plans. As a result, we believe that there is a substantial class of problems (i.e., those addressed by reactive planning) that are amenable to techniques of reactive recognition.

2 MEANS-END PLAN EXECUTION AND RECOGNITION

In this section, we illustrate informally our approach to the processes of plan execution and plan recognition using a well known example from the literature [9]. Figure 1 shows a number of plans, at different levels of granularity, to make two types of pasta dish. The BNF syntax of the plans is a simplified form of what is described in the Procedural Reasoning System (PRS) [5; 8] and is given in Figure 2.

A plan has a name, an invocation condition that can trigger the plan, a precondition that needs to be true before the plan body is started, a postcondition that is true after the plan body is finished successfully and the body of a plan which is an AND-OR acyclic graph with the edges labelled with certain plan expressions. Furthermore, we assume that the plans are non-recursive. In the BNF syntax an OR-node is represented as $-(<\text{label}>)\rightarrow$ and an AND-node as $+(<\text{label}>)\rightarrow$. For a given proposition $\alpha$, the expression $(!\alpha)$ means achieve (or recognize the achievement of) a state of the world where $\alpha$ is true. The expression $(e)$ means execute (or observe) the primitive plan or action $e$.

Consider first the process of plan execution where the agent initially wants is in the kitchen and wants to achieve a state in which it has made a pasta dish. To achieve this end, the agent will perform means-end reasoning and determine that two plans – namely, Make Pasta Dish 1 and Make Pasta Dish 2 – are applicable. If the agent adopts the first plan, it will want to achieve a state in which it has made ordinary pasta. To achieve this it has to adopt Make Ordinary Pasta, resulting in the agent wanting to achieve a state where it has made noodles, and so on. This process continues till the making of a pasta dish is completed or the execution fails as the agent was unable to complete one of the steps successfully.

Now consider the process of means-end plan execution in conjunction with means-end recognition. When an executing agent executes a primitive plan, the observer agent observes the (execution of the) primitive plan. While the executing agent can choose an applicable plan, once after the other, until one of them succeeds, the observing agent should attempt

---

$^3$When displayed graphically all edges from an OR-node are shown as arrows and all edges from an AND-node are shown as arrows with an arc connecting all the arrows.
<table>
<thead>
<tr>
<th>Plan Entity</th>
<th>Execution</th>
<th>Recognition</th>
<th>Success Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(e)</td>
<td>execute</td>
<td>observe</td>
<td>if e succeeds</td>
</tr>
<tr>
<td>(?!) with p₁...pₙ</td>
<td>sequentially run p₁ to pₙ</td>
<td>in parallel run p₁ to pₙ</td>
<td>if one of pᵢ succeeds</td>
</tr>
<tr>
<td>OR-node with l₁...lₙ</td>
<td>in parallel run l₁ to lₙ</td>
<td>in parallel run l₁ to lₙ</td>
<td>if one of lᵢ succeeds</td>
</tr>
<tr>
<td>AND-node with l₁...lₙ</td>
<td>in parallel run l₁ to lₙ</td>
<td>in parallel run l₁ to lₙ</td>
<td>if all of lᵢ succeeds</td>
</tr>
</tbody>
</table>

Table 1: Comparison of Execution and Recognition

...to recognize all the applicable plans simultaneously. Otherwise, both the executing and observing agents are performing identical operations. The correspondence between execution and recognition and the conditions under which they succeed are shown in Table 1. In this table, p₁, ... pₙ refer to the plans that can achieve α; and l₁,..., lₙ refer to the labels appearing on the outgoing edges of an OR-node or AND-node.

Now with this operational semantics let us run the example with the observer and executing agents having the same library of plans and both in the kitchen. Assume that the executing agent wants to make a pasta dish and the observer agent wants to recognize this. The executing agent can fulfill its desire by adopting either the plan Make Pasta Dish 1 or the plan Make Pasta Dish 2. However, the observer agent in order to recognize this has to adopt both these plans for recognition. This would result in the observer wanting to recognize whether ordinary pasta was being made or whether spaghetti marinara was being made. This in turn would result in the observer adopting the plans Make Ordinary Pasta and Make Spaghetti Marinara in the recognition mode. The adoption of Make Ordinary Pasta would result in the observer agent wanting to recognize the making of noodles and in turn, the adoption of the plans Make Fettucini and Make Spaghetti in the recognition mode. Assume that the executing agent adopts the plan to make ordinary pasta by first making fettucini and executes the action make-fettucini. The recognition plans Make Spaghetti Marinara and Make Pasta Dish 2 of the observer agent fail. Instead, the observer agent observes the primitive action of making fettucini and recognizes that this results in the achievement of the desire to make noodles. Also, the observer now knows that the executing agent is making an ordinary pasta (not a spaghetti marinara) and that the next step would be to make a sauce. Thus the observer decides to recognize the making of a sauce and adopts the plans of recognizing the making of alfredo sauce and the making of marinara sauce. This process continues till the entire Make Pasta Dish 1 plan is recognized.

3 ALGORITHMS

In this section, we present algorithms for means-end plan execution and recognition. We present simplified propositional versions of the algorithms without taking into account the mental attitudes (such as beliefs, desires, and intentions) of agents. We extend these algorithms to reactive planning and recognition and embed them into a BDI-interpreter [21] elsewhere [19].

The algorithm means-end-recognition (see Figure 3) takes as input a plan library given by P, a set of propositions S, and an expression E, which could be either a primitive plan or an achievement expression. The algorithm then returns a “success” or “failure” result and a set of propositions T that are true after the recognition.

If the expression we are trying to recognize is a primitive plan or action, we invoke the function observe. If the expression is an achievement expression, the Set Of Applicable
Plans (or SOAP) is computed from the given plan library $P$. A plan is said to be applicable if its invocation condition matches the incoming expression and its precondition is contained in the set $S$. Each plan in the set of applicable plans is recognized in parallel by running recognize-plan until one of them succeeds. The union of the postcondition of the plan that succeeded and the final state of the succeeding plan is given as the set of propositions $T$.

The algorithm means-end-execution is similar to the algorithm means-end-recognition except that observe and recognize-plan are replaced by execute and execute-plan, respectively. Furthermore, the applicable plans are run sequentially, one by one, until one of them succeeds. If all of them fail the execution process is said to have failed.

The functions observe and execute are primitives that return “success” or “failure” depending on the successful or failed observation or execution of an event, respectively. The assumption that the observation of events happen synchronously with the execution of actions by other agents is built into the function observe. In other words, if the event occurs before or after the agent runs observe the event will not be observed.

The algorithm for recognize-plan (see Figure 4), given a plan body, repeatedly recognizes the OR-nodes and AND-nodes of the plan body until the END node is reached or one of the nodes fails. The initial state $S$ gets continuously updated during the recognition of OR-nodes and AND-nodes.

Given a node, the algorithm for recognizing an OR-node performs means-end-recognition in parallel on all the plan expressions labelling the out-going arcs of the OR-node (see Figure 5). As soon as one of these recognitions is successful it returns with the next node to recognize. The algorithm for recognizing an AND-node performs means-end-recognition in parallel for each out-going arc (see Figure 6). If any one of these recognitions fail, the algorithm returns a failure. If all these recognitions succeed, the algorithm returns with the union of all the output sets $T$.

The algorithm for executing a plan is very similar, except that the recognitions of OR-nodes and AND-nodes is replaced by the execution of OR-nodes and AND-nodes. Algorithms for executing OR-nodes and AND-nodes are similar to the algorithms for recognizing OR-nodes and AND-nodes, except that means-end-execution replaces means-end-recognition.

Now let us go back to the example considered earlier. Let us assume that the observing agent wants to recognize the making of a pasta dish. This corresponds to the means-end-recognition algorithm being called with the expression: $(! (made-pasta-dish))$. This results in the plans Make Pasta Dish 1 and Make Pasta Dish 2 being added to the SOAP. Two instances of the algorithm recognize-plan with the plan bodies of Make Pasta Dish 1 and Make Pasta Dish 2 are set up in parallel. The plan for recognizing Make Pasta Dish 1 results in the means-end-recognition algorithm being invoked with the expression: $(! (made-ordinary-pasta))$, and so on. The calling of the various algorithms can be drawn graphically as shown in Figure 7, which shows the state of the agent just before the observation of events.

Now, if the agent observes (make-fettucini), i.e., the observe function succeeds with the (make-fettucini) event, the nodes marked (make-spaghetti) under Make Spaghetti and Make Spaghetti Marinara will fail. The expression $(! (made-noodles))$ will succeed because it is sufficient for one of the plans to succeed for the achievement expression to succeed. This will result in the next step of the plan Make Ordinary Pasta being invoked resulting in the achievement expression $(! (made-sauce))$. This will result in the two plans Make Marinara Sauce and Make Alfredo being called resulting in calls to observe make-marinara and make-alfredo. As the (make-spaghetti) under Make Spaghetti Marinara failed, the expression $(! (made-spaghetti-marinara))$ and the plan Make Pasta Dish 2 will fail. This will result in the observing agent inferring that the agent is not making a spaghetti marinara.
and is possibly making a fettucini alfredo. If the agent subsequently observes *make-alfredo* and the act *boil* it can conclude that the agent used the plan *Make Pasta Dish 1* to make an ordinary pasta; namely, a fettucini alfredo.

4 DYNAMIC AGENT LOGIC

There are two main approaches to reasoning about programs using modal logics in the Theoretical Computer Science literature: the *exogenous* and *endogenous* approaches [12]. Dynamic logic is an *exogenous* logic as it explicitly represents programs in the language. As opposed to this, *endogenous* logics such as Computation Tree Logic (CTL) and CTL* [4] do not represent programs explicitly but consider them as part of the structure over which the logic is interpreted. In theoretical computer science, dynamic logic has essentially been superseded by the endogenous logics (particularly CTL* and a number of variants of it).

We have chosen an exogenous logic (i.e., dynamic logic) to represent the plans of an agent as it fits naturally with the compositional nature of plans. When one reasons about the mental state of an agent during the execution/recognition of a plan, an endogenous logic is more appropriate. Elsewhere [22] we have developed endogenous logics CTL_{BDI} and CTL*_{BDI} to represent the mental state of an agent that captures the agent’s beliefs, desires, and intentions. Rational agents have to deal with mental states as well as plans and a combination of exogenous and endogenous logics would be ideal for these purposes. Mu-calculus [11], a generalization of CTL* and Dynamic logic, could serve this purpose.

Dynamic Logic (DL) first used for providing semantics for programming languages [17] has also been used as the basis for a logic of action [23]. We extend dynamic logic in three ways. First, we provide the semantics of plans from an internal agent viewpoint rather than from an external observer viewpoint as is usually done with dynamic logic. Second, we introduce the notion of recognition as a first class entity. As a result, agents not only have the choice to *execute* an action, they also have the choice to *observe* an action. Third, we allow the indirect call of plans facilitating means-end reasoning — a notion central to means-end plan execution and recognition. Agents must be capable of reasoning about the achievement of certain states of the world (ends) without necessarily reasoning about the programs (means) that achieve these states of the world.

In the following, we extend dynamic logic by explicitly introducing agents and providing a semantics based on an internal agent viewpoint. This logic, called *Dynamic Agent Logic (DAL)*, is better suited to reasoning about plans than dynamic logic.

4.1 SYNTAX

Consider a language $\mathcal{L}_0$ with a set of primitive propositions $PrimProp$, a set of primitive plans or actions $PrimPlans$ and a set of agents $A$. The propositional operators $\lor$ and $\neg$ are used to form *propositions*, denoted by $\alpha$, $\alpha_1$, ..., $\beta$, $\beta_1$, .... The plan operators $;\ (\text{sequence})$, $|\ (\text{non-deterministic or})$, and $||\ (\text{parallel})$, are used to form *plan expressions*, denoted by $\pi$, $\pi_1$, .... The mixed operator $!(\text{achieve})$ is used to convert propositions into plan expressions. The mixed operators $\langle\rangle\ (\text{there exists an execution})$, $\langle\langle\rangle\ (\text{for all executions})$, $\langle\langle\rangle\ (\text{there exists a recognition})$, $\langle\langle\rangle\ (\text{for all recognitions})$, are used to form *dynamic propositions*, denoted by $\phi$, $\phi_1$, ....

The set of well-formed propositions, plan expressions, and dynamic propositions are defined by the following BNF notation:

- $\alpha ::= p \mid \alpha_1 \lor \alpha_2 \mid \neg \alpha$
- $\pi ::= \beta \mid \alpha \mid \pi_1 \parallel \pi_2 \mid (\pi_1 \parallel \pi_2) \mid (\pi_1 ; \pi_2)$.
\( \phi := \alpha | \phi_1(\pi) \phi_2 | \phi_1[\pi] \phi_2 | \phi_1(\langle \pi \rangle) \phi_2 | \phi_1(\llbracket \pi \rrbracket) \phi_2 \)

In the above notation \( p \) is a primitive proposition, \( e \) is an action, and \( a \) is an agent.\(^4\)

In Section 2, we saw that a plan involved a name, an invocation condition, a precondition, a postcondition, and a body, which was an AND-OR graph with the edges of the graph labelled with simple plan expressions. Formally, we define a plan to be a tuple of the form \((!\alpha, \alpha_1, \pi, \beta_1)\) where \(!\alpha\) is the invocation condition, \(\alpha_1\) is the precondition, \(\pi\) is the body of a plan expressed as a plan expression, and \(\beta_1\) is a postcondition.

### 4.2 SEMANTICS

The semantics of dynamic logic is defined in terms of a set of states, say \( S \), and a state transition function that maps programs (both primitive and non-primitive) to a set of pairs of states, i.e., \( \mathcal{R} : 2^S \to 2^S \otimes 2^S \) and a truth assignment function, say \( L \). The formula \( \langle \pi \rangle \phi_2 \) is satisfiable in a state \( t \) iff there exists \(( t u ) \in \mathcal{R}(\pi)\) and \( \phi_2 \) is true in \( u \). Treating \( \pi \) as a program (rather than as a plan), this semantics is reasonable as no matter which process executes the program \( \pi \) the result should be the same. However, in the case of plans, the notion of agency — that is, which agent executes the plan — can make an important difference. Although the plans may be identical the capabilities [24] of agents may vary significantly leading to different end results.

Unlike the semantics of dynamic logic, we introduce a subjective view of the world by adopting a possible worlds semantics.\(^5\) Under this view there are multiple worlds each consisting of a set of states. The actions executed by an agent in a world at a particular state is given by a choice relation. A composition of such choice relations results in the execution of a plan by an agent. The actions observable by an agent in a world at a particular state is given by an observe relation. A composition of such observe relations results in the recognition of a plan by an agent.

More formally we define a possible-worlds structure \( M \) to be a tuple, \( M = (W, P, \{S_w : w \in W\}, \{\mathcal{C}_w^a(e) : w \in W, a \in A, \text{ and } e \in \text{PrimPlans}\}, \{\mathcal{O}_w^a(e) : w \in W, a \in A, \text{ and } e \in \text{PrimPlans}\}, L) \) where \( W \) is a set of worlds; \( P \) is a set of plans; \( A \) is a set of agents; for each world \( w \), \( S_w \) is a set of states; for each world \( w \), each agent \( a \), and each primitive plan \( e \) is a choice (observe) relation \( \mathcal{C}_w^a(e) \) \( (\mathcal{O}_w^a(e)) \subseteq S_w \times S_w \); and \( L \) is the truth assignment function that assigns to each state in \( w \) a set of propositional formulas, i.e., \( L_w : S_w \rightarrow 2^{\text{PrimProp}} \). Associated with each world \( w \) and plan \( \pi \) we also define the derived relations \( \mathcal{E}_w(\pi) \subseteq S_w \times S_w \) and \( \mathcal{R}_w(\pi) \subseteq S_w \times S_w \). These transition relations correspond to executions and recognitions which are a composition of various choice and observe relations, respectively.

With these preliminaries we are now in a position to define the semantics of DAL. We define the semantics only for execution and recognition formulas. The semantics of other propositional formulas is straightforward.

\( M, w_t \models \phi_1(\pi) \phi_2 \) iff \( M, w_t \models \phi_1 \) then there exists \( u \in S_w \) such that \( t \mathcal{E}_w(\pi) u \) and \( M, w_u \models \phi_2 \).

\( M, w_t \models \phi_1 (\langle \pi \rangle) \phi_2 \) iff

(a) \( M, w_t \models \phi_1 (\pi) \phi_2 \); and

(b) if \( M, w_t \models \phi_1 \) then there exists \( u \in S_w \) such that \( t \mathcal{R}_w(\pi) u \).

\(^4\)The formula \( \phi_1 (\langle \pi \rangle) \phi_2 \) (and other variants) are similar to Hoare triples and is merely a syntactic sugar for testing the truth of \( \phi_1 \) followed by \( \langle \pi \rangle \phi_2 \) as in normal dynamic logics. For convenience we shall abbreviate \( \text{true}(\pi) \phi_2 \) to \( \langle \pi \rangle \phi_2 \), \( \phi_1 (\pi) \text{true} \) to \( \phi_1 (\langle \pi \rangle) \text{true} \), and \( \text{true}(\langle \pi \rangle) \text{true} \) to \( \langle \pi \rangle \).

\(^5\)This view is more important when we discuss the mental attitudes of agents performing executions and recognitions (see Section 5).
$M, w_t \models \phi_1[\pi]\phi_2$ iff if $M, w_t \models \phi_1$ then for all $u \in S_w$ such that $tE_w(\pi)u$, $M, w_u \models \phi_2$.

In dynamic logic $[\pi]\phi$ is usually defined as $\neg(\langle \pi \rangle \neg \phi)$. As a result, unlike $\langle \pi \rangle \phi$, $[\pi]\phi$ does not imply that $\pi$ terminates. We avoid this complication by defining $[\pi]$ independently and by requiring that all computations of $\pi$ terminate.

Recognition of a plan can take place only if someone is executing a plan — in a world devoid of any event occurrences there is nothing to observe or recognize. Also there needs to be an agent who is observing the events for recognition to take place. The semantic definition of $\phi_1[\langle \pi \rangle]\phi_2$ captures these two conditions. The formula $\phi_1[\langle \pi \rangle]\phi_2$ is defined analogously.

The semantics of the various plan expressions are given by defining the transition relations $E_w$ and $R_w$. For primitive plans or actions the transition relations are directly given by the choice and observe functions. For plan expressions of the form $!\alpha$ we first look for all plans in $P$ whose invocation condition is $!\alpha$. The transition relations for $!\alpha$ is the set of all transitions such that the precondition of such a plan is satisfied in the initial state, there is a transition for the body of the plan expression, and at the end of such a transition the postcondition is satisfied. The transition relation for sequences is the concatenation of the transition relations for the individual plan expressions. The transition relation for non-deterministic OR is the union of the transition relations for the individual plan expressions. The transition relation for the parallel operator is the intersection of the transition relations of the individual plan expressions, if the intersection is non-null; otherwise it is undefined.

More formally, we have the following definitions for the transition relations. In the following definitions $T_w$ stands for either $E_w$ or $R_w$.

$E_w(a; e) = O_w^a(e)$ for a primitive plan $e$.
$R_w(a; e) = O_w^a(e)$ for a primitive plan $e$.
$T_w(!\alpha) = \{ (s, t) \mid (a) \ (s, \alpha, \delta) \in P; (b) \ M, w_s \models \alpha; (c) \ \text{there exists } t \in S_w \text{ such that } sT_w(\delta) \ t \text{ and} (d) \ M, w_t \models \beta \}$
$T_w(\pi_1 ; \pi_2) = T_w(\pi_1) \cap T_w(\pi_2) = \{ (s, u) \mid \text{there exists } t \in S_w \text{ such that } sT_w(\pi_1) \ t \text{ and there exists } u \in S_w \text{ such that } tT_w(\pi_2) \ u \}$
$T_w(\pi_1 | \pi_2) = T_w(\pi_1) \cup T_w(\pi_2)$
$T_w(\pi_1) \parallel \pi_2 = \begin{cases} T_w(\pi_1) \cap T_w(\pi_2) & \text{ if } T_w(\pi_1) \cap T_w(\pi_2) \neq \emptyset \\ \emptyset & \text{ otherwise} \end{cases}$

The model-theoretic semantics of execution and recognition is identical except for primitive plans. This is because, except for the primitive plans, there is no difference in the way the plans are used by an agent. The semantics of $|$ and $\parallel$ reflect the operational semantics (given by the algorithms) for OR-nodes and AND-nodes. Note that the success condition for OR-nodes is identical for both execution and recognition, and similarly for AND-nodes.

The axiomatization\(^6\) for DAL is given below with the modal operator $M$ denoting $[\langle \pi \rangle]$, $[\parallel \pi \parallel]$, $[\pi]$, and $\langle \pi \rangle$:

1. $\phi_1M(\pi)\phi_2 \equiv \phi_1 \supset (M(\pi) \land \phi_2)$;
2. $M(\pi)\phi \land M(\pi)\psi \supset M(\pi)(\phi \land \psi)$;

\(^6\)The axiom and inference rules are not the minimal set; we have included some of the theorems of the system as axioms for the purposes of clarity.
3. $M(\pi)(\phi \lor \psi) \equiv M(\pi)\phi \lor M(\pi)\psi$;
4. $M(\pi_1 | \pi_2)\phi \equiv M(\pi_1)\phi \lor M(\pi_2)\phi$;
5. $M(\pi_1 \parallel \pi_2)\phi \equiv M(\pi_1)\phi \land M(\pi_2)\phi$;
6. $M(\pi_1 ; \pi_2)\phi \equiv M(\pi_1)M(\pi_2)\phi$;
7. $\phi_1[\pi]\phi_2 \supset \phi_1(\langle \pi \rangle)\phi_2$;
8. $\phi_1[\pi]\phi_2 \supset \phi_1(\pi)\phi_2$;
9. $\phi_1[\pi]\phi_2 \supset \phi_1(\pi)\phi_2$;
10. $\phi_1(\langle \pi \rangle)\phi_2 \supset \phi_1(\pi)\phi_2$;
11. Modus Ponens.
12. Modal Generalization: From $\vdash \phi$ infer $\vdash M(\pi)\phi$.
13. Achievement Plans (1&2): From a given set of plans $P$ infer $P \vdash [\alpha] \equiv \bigwedge_{(\alpha,\alpha_i,\delta_i,\beta_i) \in P} \alpha_i[\delta_i]\beta_i$; and similarly for $[]$.
14. Achievement Plans (3&4): From a given set of plans $P$ infer $P \vdash \langle\alpha\rangle \equiv \bigvee_{(\alpha,\alpha_i,\delta_i,\beta_i) \in P} \alpha_i(\delta_i)\beta_i$; and similarly for $\langle\rangle$.

The first six axioms and the inference rule Modal Generalization are common to all the four modal operators. Except for $[]$, the axioms are similar to those of dynamic logic. Dynamic logic does not have the $\parallel$ operator. However, dynamic logic has a test operator (?) and an iteration operator ($\ast$) which we have omitted here for simplicity.\(^7\)

Axioms (7)-(10) are multi-modal axioms that link the various execution and recognition operators. The inference rules for achievement of states connects the achievement expression with a plan that achieves the state. In the case of all recognitions (executions) we require that all the plans that achieve the state be recognized (executed) and in the case of a single recognition (execution) we just require one of the plans that achieve the state to be recognized.

Note that we have associated agency only with respect to an agent’s actions (i.e., primitive plans). Extending the syntax and semantics so as to associate the notion of agency to non-primitive plans is trivial, if all the actions are performed by the same agent. However, extending this choice to plans which involve actions by other agents is a non-trivial task. This is because it is not clear what it means for an agent $a$ to have the choice of executing a plan that involves some other agent $b$ executing an action. One possible interpretation is for agent $a$ to send a message to $b$ to execute its action and wait for a successful completion of that action. An operational semantics along these lines was discussed elsewhere [10]. However, giving a formal account of such a theory would involve extending dynamic logic with the message passing paradigm of CSP [7] and is beyond the scope of this paper.

\(^7\)Introducing the iteration operator will allow us to permit cyclic and recursive AND-OR graphs as plan bodies. As the iteration operator $\pi^*$ is a nondeterministically chosen finite number of iterations of $\pi$ we cannot have infinite loops in the plan.
Now we consider the relationship between the algorithms for plan execution and recognition introduced in Section 2 and the dynamic agent logic. In particular, we want to establish the relationship between the successful running of the algorithm \textit{means-end-recognition} and the provability of recognition formulas in the dynamic agent logic. As a corollary, we get the corresponding relationship for means-end plan execution.

First, we convert the AND-OR plan graphs discussed in Section 2 into DAL expressions. The primitive plan \((e)\) is equivalent to \((a:e)\) with \(a\) being the agent executing/observing \(e\). The plan fragment with two adjacent arcs labelled \(l_1\) and \(l_2\) is equivalent to \(l_1; l_2\). If \(l_1 \ldots l_n\) are the labels on outgoing edges of an OR-node, they are equivalent to \(l_1 | l_2 | \ldots | l_n\). Similarly, if \(l_1 \ldots l_n\) are the labels on outgoing edges of an AND-node, they are equivalent to \(l_1 \land l_2 \land \ldots \land l_n\). From these basic transformations one can easily convert the body of a plan into a single plan expression \(\delta\). A plan with invocation condition \((!a)\), precondition \(\alpha_1\), postcondition \(\beta_1\), and a body whose equivalent plan expression is \(\delta\), is treated as a formal plan \((!a, \alpha_1, \delta, \beta_1)\). For example, the plan \textit{Make Ordinary Pasta} is formally equivalent to \(((! (\text{made-ordinary-pasta})), \text{true}, ((! (\text{made-noodles}) \land (! (\text{made-sauce}))) \land (! (\text{a:boil}))), \text{true})\).

Having converted AND-OR plan executions/recognitions into equivalent DAL formulas we now examine the successful running of the algorithms with the provability of certain formulas.

Consider the simple case where the means-end-recognition algorithm for agent \(a\) is given a set of plans \(II\), a set of propositions \(\Gamma\), and an expression of the form \(e\) where \(e\) is a primitive plan. If the algorithm returns successfully with a set of propositions \(\Lambda\), it would be reasonable for us to assume that the equivalent recognition expression \(\gamma \langle \langle a:e \rangle \rangle \lambda\) is valid in the dynamic agent logic, where \(\gamma\) and \(\lambda\) are conjunctions of propositions in \(\Gamma\) and \(\Lambda\). Note that a successful run of the algorithm corresponds to there being at least one recognition path (rather than all recognition paths) where \(e\) is observed. If in fact the stronger recognition formula, namely \(\gamma \langle \langle a:e \rangle \rangle \lambda\) is valid in the dynamic agent logic we can state that the means-end recognition algorithm \textit{will} succeed with the input set of propositions \(\Gamma\) and output set of propositions \(\Lambda\). Formally, we have the following proposition:

\textbf{Proposition 1} (a) If \textit{means-end-recognition}(\(II\), \(\Gamma\), \(e\), \(\Lambda\)) returns “success” for agent \(a\) then \(\vdash \gamma \langle \langle a:e \rangle \rangle \lambda\).

(b) If \(\vdash \gamma \langle \langle a:e \rangle \rangle \lambda\) then \textit{means-end-recognition}(\(II\), \(\Gamma\), \(e\), \(\Lambda\)) returns “success” for agent \(a\).

In the above cases \(\gamma\) and \(\lambda\) are the conjunction of all propositions in \(\Gamma\) and \(\Lambda\), respectively.

From the above proposition we can show that similar results hold for executions.

Now let us consider expressions of the form \((! \phi)\). We want to show that if the means-end-recognition algorithm is called with a set of plans \(II\), set of input propositions \(\Gamma\) and the expression \((! \phi)\), and succeeds with the output set of propositions \(\Lambda\) then the corresponding recognition formula \(\gamma \langle \langle ! \phi \rangle \rangle \lambda\) is provable with respect to the set of plans \(II\).

The first step in this proof involves transforming each plan in the plan library into its formal equivalent as detailed above. This step also converts the body of plans into an equivalent plan expression. Next we need to show that each step of the algorithm (i.e., observing actions, recognizing plans, recognizing OR-nodes, and recognizing AND-nodes), corresponds to the axioms (or semantic definitions) of DAL; that is, observing primitive actions (Proposition 1 above), axiom for ; and inference rule for \(! \alpha\), axiom for \(|\), and axiom for ||, respectively. More formally we can state the following theorem:

\textbf{Theorem 1} If \textit{running} \textit{means-end-recognition}(\(II\), \(\Gamma\), \((! \alpha)\), \(\Lambda\)) by agent \(a\) returns “success” then \(II \vdash \gamma \langle \langle ! \alpha \rangle \rangle \lambda\), where \(\gamma\) and \(\lambda\) are conjunctions of propositions in \(\Gamma\) and \(\Lambda\), respectively.
Proof: The proof involves a case by case analysis of all the algorithms.

means-end-recognition: If the input expression is an achievement expression of the form $!\alpha$ then this algorithm first computes the set of applicable plans. The invocation condition of plans of this set is $!\alpha$ and the precondition of these plans is contained in the input set of formulas $\Gamma$. This is equivalent to stating that $\Pi \vdash \forall \alpha_1$ and so on till $\Pi \vdash \forall \alpha_n$, where $\alpha_1 \ldots \alpha_n$ are such that they are the preconditions of formal plans whose invocation is $!\alpha$.

Next the algorithm calls recognize-plan for each one of the plan bodies of the set of applicable plans. Let the plan expressions corresponding to these plan bodies be $\delta_1 \ldots \delta_n$. As the entire means-end-recognition algorithm succeeds (by the premise of the theorem) at least one of these plan bodies, say $\delta_i$ will succeed. Assuming that recognizing the plan body is equivalent to $\forall \langle \delta_i \rangle \tau_i$ where $\tau_i$ is the conjunction of $T_i$, at the end of recognizing the plan we have $\Pi \vdash \forall \alpha_i \vdash \forall \langle \delta_i \rangle \tau_i$. Note that we have written $\forall \alpha_i \vdash \forall \langle \delta_i \rangle \beta_i$ because, if the preCondition is not satisfied the plan body cannot be run.

The step that adds the postcondition of the plan results in $\Pi \vdash \lambda \equiv \tau_i \wedge \beta_i$. Continuing on from the previous paragraph we also have $\Pi \vdash \forall \alpha_i \vdash \forall \langle \delta_i \rangle (\beta_i \wedge \tau_i)$. From the inference rule for Achievement Plans 3 we have $\Pi \vdash \forall \langle \lambda \rangle \equiv \alpha_i \vdash \forall \langle \delta_i \rangle \beta_i$. Hence, we have $\Pi \vdash \forall \langle \lambda \rangle \equiv \forall \langle \lambda \rangle \lambda$. This is equivalent to $\Pi \vdash \forall \langle \lambda \rangle \lambda$.

Now we prove our assumption that recognizing the plan body is equivalent to $\forall \langle \delta_i \rangle \tau_i$.

recognize-plan: Let the plan expression corresponding to a plan body $\delta_i$ be a sequence of $\delta_{i1} \ldots \delta_{ij}$ plan expressions, i.e., $\delta_i \equiv \delta_{i1} \ldots \delta_{ij}$. Each one of the $\delta_{im}$ where $m = 1$ to $j$ corresponds to a plan expression equivalent to the labelling of the outgoing arcs of the node $n$. The node can either be an OR-node or an AND node. Let us assume that recognizing an OR-node or AND-node, $m$, is equivalent to $\Pi \vdash \sigma_{im} \forall \langle \delta_{im} \rangle \sigma_{im+1}$, where $\sigma_{im}$ is the state of the world before recognizing $\delta_{im}$ and $\sigma_{im+1}$ is the state of the world after the recognition.

By the first step of the plan $\gamma \equiv \forall \delta_{i1}$. By our assumption above we have $\Pi \vdash \forall \delta_{i1} \forall \langle \delta_{i1} \rangle \sigma_{i2}$, when the OR-node or AND-node corresponding to $\delta_{i1}$ succeeds. When we go through the loop in the recognize-plan for the second time $S_i$ now is $\sigma_{i2}$. By assumption above we have $\Pi \vdash \forall \delta_{i2} \forall \langle \delta_{i2} \rangle \sigma_{i3}$. Combining these two, we have $\Pi \vdash \forall \delta_{i1} \forall \langle \delta_{i1} \rangle \forall \delta_{i2} \forall \langle \delta_{i2} \rangle \forall \delta_{i3}$. From Axiom 1 for $\langle \rangle$ and propositional axioms we have $\Pi \vdash \forall \delta_{i1} \forall \delta_{i2} \forall \langle \delta_{i2} \rangle \forall \delta_{i3}$. From the axiom for sequencing (Axiom 6) we have $\Pi \vdash \forall \delta_{i1} \forall \delta_{i2} \forall \langle \delta_{i2} \rangle \forall \delta_{i3}$. This process can be continued for $j$ steps of the loop, where $j+1$ is the number of nodes of the plan. At the end of this we will have $\Pi \vdash \forall \delta_{i1} \forall \delta_{ij} \forall \langle \delta_{ij} \rangle \forall \delta_{i+1}$. Given that the sequence of $\delta_{i1}, \ldots, \delta_{ij}$ was equivalent to $\delta_i$ we have $\Pi \vdash \forall \delta_{i1} \forall \delta_{ij} \forall \langle \delta_{ij} \rangle \forall \delta_{i+1}$. From the equivalences $\gamma \equiv \forall \delta_{i1}$ and $\tau_i \equiv \forall \delta_{i+1}$, we have $\Pi \vdash \forall \langle \delta_i \rangle \tau_i$.

Now we prove that recognizing an OR-node, $m$, is equivalent to $\Pi \vdash \forall \delta_{im} \forall \langle \delta_{im} \rangle \forall \delta_{im+1}$, where $\sigma_{im}$ is the state of the world before recognizing $\delta_{im}$ and $\sigma_{im+1}$ is the state of the world after the recognition.

recognize-OR-node: Let plan expression $\delta_{im} \equiv \delta_{i1} \ldots \delta_{ik}$, where $k$ is the number of outgoing arcs from node $n$. Each one of these $\delta_{im1}$ to $\delta_{imk}$ is either a primitive action or an achievement formula of the form $!\alpha$. The call to the means-end-recognition results in checking the provability or otherwise of $\sigma_{im} \forall \langle \delta_{im1} \rangle \sigma_{im+1} \ldots \sigma_{imk} \forall \langle \delta_{imk} \rangle \sigma_{im+1}$. As the algorithm recognize-OR-node succeeds, at least one of the arcs, say $\delta_{imn}$ where $n$ is one of $1..k$ should succeed.

If $\delta_{imn} \equiv e$ then by Proposition 1 we have $\Pi \vdash \forall \delta_{im} \forall \langle e \rangle \forall \delta_{im+1}$. If $\delta_{imn} \equiv !\alpha'$ then invoking the current theorem for $!\alpha'$ results in $\Pi \vdash \forall \delta_{im} \forall \langle !\alpha' \rangle \sigma_{im+1}$ and $\sigma_{im+1} \equiv \sigma_{im+1}$. In either case, we have $\Pi \vdash \forall \delta_{im} \forall \langle \delta_{imn} \rangle \sigma_{im+1}$. We can replace $\sigma_{im+1}$ by $\forall \delta_{im+1}$ and add disjuncts for all the other plan expressions to obtain $\Pi \vdash \forall \delta_{im} \forall \langle \delta_{im1} \rangle \forall \delta_{im+1}$ $\forall \delta_{im+1} \forall \delta_{im+1} \forall \delta_{im+1} \forall \delta_{im+1}$ $\forall \delta_{im+1} \forall \delta_{im+1} \forall \delta_{im+1}$. From the axiom for non-deterministic or recognition (Ax
iom 4) we have, $\Pi \vdash \sigma_{im} \langle \hat{\epsilon}_{im1}, \ldots, \hat{\epsilon}_{imk} \rangle \sigma_{im+1}$, where $\sigma_{im+1} \equiv \sigma_{i(m+1)}^1 \lor \cdots \lor \sigma_{i(m+1)}^k$. This is equivalent to $\Pi \vdash \sigma_{im} \langle \hat{\epsilon}_{im} \rangle \sigma_{im+1}$.

**recognize-AND-node:** The reasoning for the AND-node proceeds in a similar manner to that of **recognize-OR-node**, except that all the out-going arcs of the AND node succeed. Hence we have $\Pi \vdash \sigma_{im} \langle \hat{\epsilon}_{im1}, \ldots, \hat{\epsilon}_{imk} \rangle \sigma_{i(m+1)}^i$ and so on until $\Pi \vdash \sigma_{im} \langle \hat{\epsilon}_{im} \rangle \sigma_{i(m+1)}^k$. By the axiom for $\langle \rangle$ and propositional axioms we can write this as $\Pi \vdash \sigma_{im} \supset (\langle \hat{\epsilon}_{im1} \rangle \land \sigma_{i(m+1)}^1 \land \cdots \land \langle \hat{\epsilon}_{imk} \rangle \land \sigma_{i(m+1)}^k)$. Once again by propositional rearrangement this is equivalent to $\Pi \vdash \sigma_{im} \supset (((\langle \hat{\epsilon}_{im1} \rangle \land \sigma_{i(m+1)}^1 \land \cdots \land \sigma_{i(m+1)}^k) \land \cdots \land (\langle \hat{\epsilon}_{imk} \rangle \land \sigma_{i(m+1)}^1 \land \cdots \land \sigma_{i(m+1)}^k) \land \cdots \land (\langle \hat{\epsilon}_{imk} \rangle \land \sigma_{i(m+1)}^1 \land \cdots \land \sigma_{i(m+1)}^k) \cdots \land \langle \hat{\epsilon}_{im} \rangle \land \sigma_{i(m+1)}^k) \land \cdots \land (\langle \hat{\epsilon}_{im} \rangle \land \sigma_{i(m+1)}^k) \land \cdots \land (\langle \hat{\epsilon}_{im} \rangle \land \sigma_{i(m+1)}^k))$. From the axiom for $\|$, for recognition (Axiom 5) we have $\Pi \vdash \sigma_{im} \supset (((\langle \hat{\epsilon}_{im1} \rangle \land \cdots \land \langle \hat{\epsilon}_{imk} \rangle \land \sigma_{i(m+1)}^1 \land \cdots \land \sigma_{i(m+1)}^k) \cdots \land (\langle \hat{\epsilon}_{imk} \rangle \land \sigma_{i(m+1)}^1 \land \cdots \land \sigma_{i(m+1)}^k) \cdots \land (\langle \hat{\epsilon}_{imk} \rangle \land \sigma_{i(m+1)}^1 \land \cdots \land \sigma_{i(m+1)}^k))$. Replacing $\hat{\epsilon}_{im}$ and from the axiom for $\langle \rangle$(Axiom 1), we finally have $\Pi \vdash \sigma_{im} \langle \hat{\epsilon}_{im} \rangle \sigma_{im+1}$, where $\sigma_{im+1} \equiv \sigma_{i(m+1)}^1 \land \cdots \land \sigma_{i(m+1)}^k$. ♦

The converse of Theorem 1 is false because there can be recognition paths that fail to recognize $\alpha$ from the given set of formulas $\Gamma$. However, strengthening the consequent of the above theorem we can state that if it is provable from $\Pi$ that $(! \alpha)$ succeeds in all recognition paths then **means-end-recognition** will return “success”. More formally we have the following theorem:

**Theorem 2** If $\Pi \vdash \gamma([! \alpha]) \lambda$, then running means-end-recognition($\Pi, \Gamma, ([! \alpha]), \Lambda$) by agent a will return “success”, where $\gamma$ and $\lambda$ are conjunctions of propositions in $\Gamma$ and $\Lambda$, respectively.

**Proof:** The proof of this theorem is similar to that of Theorem 1 except for two major differences:

- all the axioms used are that of $[\|]$, rather than $\langle \rangle$; and
- the proof proceeds in a bottom-up fashion, i.e., we first show that if $\Pi \vdash \gamma(\langle \hat{\epsilon}_{im} \rangle \lambda)$, where $\hat{\epsilon}_{im}$ is a plan expression with all parallel operators then an AND-node that is labelled by the same expression should succeed with the correct input and output arguments. We progressively work ourselves from recognizing OR and AND-nodes, to recognizing a plan, to a means-end reasoner. ♦

These two theorems establish a strong relationship between the means-end-recognition algorithm and the recognition formulas of DAL. They provide the relative soundness and completeness of the means-end-recognition algorithms with respect to a given set of plans $\Pi$.

As corollaries of these two theorems we also obtain similar relationships between running the algorithm means-end-execution and the execution formulas of DAL.

Going back to our example consider the state of the world where the executing agent has just executed the primitive plan *make-fettucini*. The means-end-recognition for the observer would succeed for *make-fettucini*. The corresponding recognition formula, in-kitchen$\langle$*make-fettucini*$\rangle$ will be true in DAL. Now from the inference rule for achievement plans, given the set of plans $\Pi$, for making and recognizing pastas we know that *make-fettucini* is one way of achieving *made-noodles*. Therefore the agent would recognize the achievement of *made-noodles* and hence the formula in-kitchen$\langle$!(*made-noodles*)$\rangle$ will be true in DAL.

Independent of the algorithms for means-end-recognition and the above theorems we can prove from the axioms and inference rules of DAL that a given set of plans $P$ (as in Figure 1), a set of observations $O$, which includes the dynamic formula $\langle$*make-fettucini*$\rangle$, and a set of propositions $\Gamma$ that includes in-kitchen, we can prove in DAL that $\langle$!(*made-noodles*)$\rangle$. In other words, $P \cup O \cup \Gamma \vdash \langle$!(*made-noodles*)$\rangle$.  

11
5 REACTIVE RECOGNITION

As discussed earlier, means-end plan recognition is based on two restrictive assumptions, namely, the world does not change during recognition, and the occurrence of events in the external world is synchronous with the recognition of events by the agent. Now we discuss modifications required to the algorithms for removing these assumptions.

To notice changes occurring in the environment during the process of recognition, the recognition algorithm has to return control to the main loop after every step of the plan. The main loop can then decide based on the new information from the environment if it is rational to proceed with the plan it is currently running or change its focus and invoke a new plan. This can be done only if the run state of the plan is captured from one step of the plan to the other, i.e., between the interrupts from the environment.

Capturing the state of a plan which is partially run and continuing to run it as long as there is no significant change in the environment introduces the notion of a commitment towards a plan. Such a commitment by an agent towards a plan is called an intention. The agent invokes a plan to satisfy a certain desire, i.e., the invocation condition of the plan. The precondition of the plan are what the agent should believe to be true before running the plan. Thus, we end up with a belief, desire, intention (or for short mental-state) interpretation of plan execution.

This mental-state interpretation of reactive plan execution is well known within the community [2; 21; 24]. One can provide an analogous mental-state interpretation of reactive plan recognition: if the agent acquires a desire to recognize the achievement of a certain state of the world it adopts all plans and intends to recognize all such plans; intending to recognize a plan will result in the agent adopting a desire to recognize the first arc in the body of the plan; this will in turn result in the agent adopting further intentions towards all plans that can recognize the desire. At any point in time the current recognition trace will enable the agent to infer the beliefs, desires, and intentions of other agents.

Having inferred the mental state of other agents, the agent can then base its future executions and recognitions on such inferred mental states. In other words, one can write plans whose precondition involves the beliefs, desires, and intentions of other agents, which have been inferred by the above process. This leads to a powerful model of interleaved reactive plan execution and recognition.

Also one can modify the syntax and semantics of plans so that the invocation condition captures the achievement or the recognition of the achievement of certain states explicitly. Similarly, the plan expressions labelling the edges can explicitly capture the execution/observation of primitive plans and the execution/recognition of the achievement of certain states. This would then provide a resource-bounded agent to balance its observation acts and recognition desires, with its execution acts and executional desires. In other words, the agent can deliberate on whether to sense or act and how long to sense before acting and how long to act before sensing.

We get rid of the other assumption of synchronized occurrence of events in the external world and the observation of events by making agents wait indefinitely to observe an event by suspending the corresponding intention. This models an agent with fanatical or blind commitment towards its recognition desires. A more reasonable model for an agent would be to have an open-minded or single-minded commitment [20]).

While the above commitment is the commitment of the observing agent towards its own recognitions, the observing agent may also need to assume (or better still, recognize) the type of commitment adopted by executing agents. This leads to interesting possibilities in terms of agents trying to recognize how other agents are attempting to recognize their own
actions. This information can then lead to some agents trying to deceive other agents (e.g., their opponents in adversarial domains) into believing that they are fulfilling certain desires, while in fact they are actually attempting to thwart the recognition desires of the other agents.

6 COMPARISON AND CONCLUSIONS

Regular and Context-free Languages: It is well known that a Propositional Dynamic Logic (PDL) program can be viewed as a regular expression denoting the set of its computation sequences [12]. In its simplest form the plans introduced in this paper can be viewed likewise. However, the presence of preconditions, postconditions, and the indirect call of plans (as done by the achievement operator) make the plans less like a regular expression. In reactive recognition when the preconditions can be complex modal formulas of beliefs, desires, or intentions, the plans can no longer be viewed as simple grammars for regular languages.

Allowing recursive calls of plans results in a context-free grammar. This would correspond to context-free PDL [12]. Once again with more complex preconditions these plans would no longer be equivalent to context-free grammars.

Plan Recognition: Early work by Allen and Perrault [1] and more recently by Litman and Allen [13] treat plan recognition as the reverse process of planning (in the classical sense). Litman and Allen’s work make use of a plan library with a rich hierarchical structure. However, unlike the theory outlined here, these plans are used in a bottom-up fashion to construct an explanation of observed behaviour on the basis of observed actions, rather than running the plans in a top-down fashion as done in this paper.

Kautz [9] presents a formal approach to plan recognition which makes use of an event hierarchy to guide the recognition process. An explanation (c-entailment) is constructed for each observation using the event hierarchy. Different possible explanations are combined by selecting covering models that minimize the number of events. This is done by circumscription. Kautz also provides graph-based algorithms for plan recognition.

While Kautz’s approach proceeds bottom-up, creating an explanation for each observation and then merging these explanations, the means-end plan recognition proceeds top-down by requiring the agent to specify what top-level states of the world it is expecting to recognize and then constructing the explanation incrementally guided by the plans and the observation of events. For example, in our approach the agent needs to invoke the means-end recognition algorithm with an expression such as (!made-pasta-dish) before the making of a pasta dish; otherwise the agent will not be able to recognize the making of a pasta dish, even if it had such a plan. This is not the case in Kautz’s approach.

Kautz deals with a more powerful underlying interval temporal logic compared to our state-based dynamic logic. Also, when an observation does not match what the function observe is expecting to observe the recognition plan fails. Thus, when extraneous events (i.e., events which are not associated with the current input expression) occur the means-end recognition will fail to recognize the plan, while Kautz’s algorithm is robust enough to infer such plans. The reactive recognition algorithm discussed elsewhere [19] will recognize such plans.

In spite of the above differences, our approach gives the same result as Kautz, in limited cases, provided the following assumptions are true: (a) events are observed in the order in which they are specified in the plan with no extraneous events; and (b) the agent chooses a subset of plans from its plan library to recognize and the final recognized plans fall within this subset. For reactive recognition assumption (a) is not required. Assumption (b) results
in a loss of generality, but increases efficiency, thereby making the approach feasible for resource-bounded agents situated in dynamic worlds.

Means-end plan recognition is fairly constrained compared to general plan recognition. However, when embedded within the other mental attitudes of an agent and combined with reactive planning it leads to a more powerful theory. This paper lays the foundation for such an integrated theory of reactive planning and recognition by providing theoretical principles of means-end recognition and analyzing its relationship to means-end execution. It also discusses how these principles can be embodied within existing reactive planning systems. Although a number of issues remain to be addressed within this form of reactive recognition and are the subject of future work, we feel that the approach shows promise in a large number of application domains where reactive planning has been used successfully [8; 14; 18].

In summary, the primary thrust of this paper is to shift (at least partially) the focus of attention within the plan recognition community towards reactive recognition.

Acknowledgements

I would like to thank Mike Georgeff for valuable comments on earlier drafts of this paper and the the anonymous reviewers for their useful suggestions. This research was supported by the Cooperative Research Centre for Intelligent Decision Systems under the Australian Government’s Cooperative Research Centres Program.
References


PLAN: Make Pasta Dish 1
invocation condition: (! (made-pasta-dish))
precondition: (in-kitchen)
body:
(! (made-ordinary-pasta))
postcondition: (meal-prepared)

PLAN: Make Pasta Dish 2
invocation condition: (! (made-pasta-dish))
precondition: (in-kitchen)
body:
(! (made-spaghetti-marinara))
postcondition: (meal-prepared)

PLAN: Make Ordinary Pasta
invocation condition: (! (made-ordinary-pasta))
precondition: (in-kitchen)
body:
(! (made-noodles)) (! (made-sauce)) (boil)

PLAN: Make Spaghetti Marinara
invocation condition: (! (made-spaghetti-marinara))
precondition: (in-kitchen)
body:
(make-spaghetti) (make-marinara)

PLAN: Make Fettucini
invocation condition: (! (made-noodles))
precondition: (in-kitchen)
body:
(make-fettucini)

PLAN: Make Spaghetti
invocation condition: (! (made-noodles))
precondition: (in-kitchen)
body:
(make-spaghetti)

PLAN: Make Marinara Sauce
invocation condition: (! (made-sauce))
precondition: (in-kitchen)
body:
(make-marinara)

PLAN: Make Alfredo Sauce
invocation condition: (! (made-sauce))
precondition: (in-kitchen)
body:
(make-alfredo)

Figure 1: Plan library for making pastas
<plan> ::= <name> <invocation> <precond>
     <postcond> <body>

<name> ::= string
<invocation> ::= !α
<precond>, <postcond> ::= α
<body> ::= <node> {- (<label>) →} + <body> |
     <node> {+ (<label>) →} + <body> |
     <node>
<label> ::= ε | !α
<node> ::= symbol
{a}+ stands for one or more a’s.

Figure 2: BNF syntax for Plans

procedure means-end-recognition(P, S, E, T)
case type-of(E) is
  primitive-action:
    result = observe(E);
    return(result);
  achievement:
    soap := {\{};
    for p_i in P do
      if ((E = invocation-condition(p_i) and
        (precondition(p_i) ⊆ S)) then
        SOAP := SOAP ∪ p_i;
    in parallel for each p_i in SOAP do
      result_i = recognize-plan(P, S, body(p_i), T_i);
      if (result_i = success) then
        T := T_i ∪ postcondition(p_i);
      return(success)
    return(failure).

Figure 3: Algorithm for means-end recognition
\begin{procedure}
\textbf{recognize-plan}(P, S, \text{plan-body}, T)
\begin{description}
\item n := start-node(\text{plan-body}); S_i := S;
\item while (not (end-node(n))) do
  \begin{description}
  \item case type-of(n) is
    \begin{description}
    \item OR:
      \begin{description}
      \item result := recognize-OR-node(P, S_i, n, T_i, next);
      \end{description}
    \item AND:
      \begin{description}
      \item result := recognize-AND-node(P, S_i, n, T_i, next);
      \end{description}
    \end{description}
    \end{description}
  if (result = success) then
    \begin{description}
    \item n := next-node;
    \item S_i := T_i;
    \end{description}
  else return(failure);
  \end{description}
  T := T_i;
\end{description}
return(success).
\end{procedure}

\textbf{Figure 4: Algorithm for recognizing a plan body}

\begin{procedure}
\textbf{recognize-OR-node}(P, S, n, T, next)
\begin{description}
\item in parallel for i = 1 to |out-arcs(n)| do
  \begin{description}
  \item e_i := out-arcs_i(n);
  \item result_i := means-end-recognition(P, S, label(e_i), T_i);
  \item if (result_i = success) then
    \begin{description}
    \item next := dest-node(e_i);
    \item T := T_i;
    \end{description}
  \end{description}
\end{description}
return(failure).
\end{procedure}

\textbf{Figure 5: Algorithm for recognizing OR nodes}

\begin{procedure}
\textbf{recognize-AND-node}(P, S, n, T, next)
\begin{description}
\item in parallel for i = 1 to |out-arcs(n)| do
  \begin{description}
  \item e_i := out-arcs_i(n);
  \item result_i := means-end-recognition(P, S, label(e_i), T_i);
  \item if (result_i = failure) then return(failure)
  \item T := \bigcup_{i = 1}^{|out-arcs(n)|} T_i;
  \item next := dest-node(e_i);
  \end{description}
\end{description}
return(failure).
\end{procedure}

\textbf{Figure 6: Algorithms for recognizing AND nodes}
Figure 7: Call Graph for making pastas (just before the first observation)