# Multi-Agent Mental-State Recognition and its Application to Air-Combat Modelling

Anand S. Rao
Australian Artificial Intelligence Institute
171 La Trobe Street
Melbourne, Victoria 3000
Australia
Email: anand@aaii.oz.au

Graeme Murray
Aeronautical and Maritime Research Laboratory
Defense Science and Technology Organisation
Port Melbourne, Victoria 3207
Australia

May 30, 1994

**Abstract**

Recognizing the mental-state, i.e., the beliefs, desires, plans, and intentions, of other agents situated in the environment is an important part of intelligent activity. Doing this with limited resources and in a continuously changing environment, where agents are continuously changing their mind, is a challenging task. In this paper, we provide algorithms for performing reactive plan recognition and embed it within the framework of an agent's mental-state. This results in a powerful model for mental-state recognition and integrated reactive plan execution and plan recognition. We then apply this in an adversarial domain – air-combat modelling – to enable pilots to infer the mental-state of their opponents and choose their own tactics accordingly. The entire approach is based on using plans as recipes and as mental-attitudes to guide and contrain the reasoning processes of agents.

## 1 Introduction

Agents are computational entities that are situated in dynamic environments, acting to fulfill desired ends, and reacting to changing situations. Agents or, at least, an important subclass of agents, can be viewed as having mental states that comprise the beliefs, desires, plans, and intentions both of themselves and of others. While it is reasonable for us, as designers of multi-agent systems, to provide for an agent its own beliefs, desires, and plans (based on which it forms its intentions), it is quite natural for the agents then to *recognize* the beliefs, desires, plans, and intentions of the other agents in its environment.

There has been considerable work in recent years on plan recognition [9, 11] that focuses on inferring the plans and intentions of other agents. However, most of this work treats plan recognition as the reverse process of classical planning, concerned with inferring plans in a bottom-up fashion from observations. Furthermore, this work is not integrated with the overall mental-state of the agent, i.e., its beliefs and desires.

Over the past decade the focus of research in planning has shifted from classical planning to reactive or situated planning [7]. 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. These have 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 intertwined in a complex web of relationships with other mental attitudes of beliefs, desires, and intentions [12]. 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 is done in classical planning. Plans as mental attitudes *constrain* the agent in its future decision-making by committing it to previously made rational decisions. The latter are called *intentions.*

The use of plans as recipes to guide the recognition process and the use of plans as mental attitudes to constrain the recognition process will be called *reactive recognition.* When reactive recognition leads to the recognition of beliefs, desires, plans, and intentions of other agents in the environment, we call the process *mental-state recognition.*

Mental-state recognition is applicable to a broad class of problems where: (a) agents have limited resources; and (b) the environment and the agents in such environment are changing their mental-state *while* the agents are doing their recognition. However, the process of mental-state recognition is based on two important assumptions: (a) the recognizing agent has correct and 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 the first assumption, the observer's knowledge of the plans of the executing agent *guides* the observer's recognition process. We have assumed that the oberver's model of the executing agent is sound and complete. The soundness and completeness of the recognition process is dependent on the observer's model, i.e., if the oberver's model of the executing agent is incomplete, the recognition would still occur but would be incomplete. In the second assumption the adoption of plans to recognize specific desires of the executing agent *constrains* the recognition process. In other words, the observer agent decides what desires need to be recognized and when they need to be recognized. This models *attentiveness* of agents – an agent is attentive towards an executing agent's desire only if it has decided to recognize the desire. If the observer agent is not attentive towards certain desires, these desires will not be recognized even if the executing agent executes plans to satisfy these desires.

Elsewhere [14] we presented a limited form of reactive recognition called *means-end recognition* and provided a model-theoretic and operational semantics for it. In this paper, we extend the means-end recognition process to reactive recognition by modifying an existing Belief-Desire-Intention (BDI) interpreter [17] used for reactive planning. Given that reactive recognition is taking place within the framework of an agent's other mental attitudes we can also infer the mental-state of other agents, thereby providing a theory of mental-state recognition.

Mental-state recognition is a generic process that can be applied to a number of different domains, like discourse understanding, automated tutoring, and adversarial reasoning. In this paper, we focus on an application that was the driving force for developing our theory — modelling situation awareness in a multi-aircraft air-to-air combat system.

## 2 Situation Awareness and Tactics Selection in Air Combat Modelling

A combat pilot receives large amounts of information from a variety of different sources, analyses this information, attempts to determine his current situation, selects the tactics to follow, and coordinates his actions with other pilots. All these steps have to be done continuously during the entire course of his mission and have to be done quickly.

Air combat simulation studies are undertaken to determine the parameters involved in aircraft and weapons performance, pilot reaction, and tactics adopted. Computer simulation of air combat usually[1] involves modelling both the physical aspects of the aircraft and the tactical knowledge of the pilot. The physical aspects of the aircraft include its performance determinants and its sensor capabilities. The tactical knowledge of the pilot encompasses his ability to assess the situation in which he finds himself and the selection of appropriate actions to take.

The tasks performed by a combat pilot can be broadly divided into two areas – *situation awareness* and *tactics selection.*

Situation awareness is defined as *the knowledge, understanding, cognition and anticipation of events, factors, and variables affecting the safe, expedient, and effective conduct of a mission* [8]. Situation awareness is a central issue in all computer modelling of human decision makers. Representation of situation awareness in air combat, an application in which time constraints on decision making can be extremely tight, is a considerable modelling challenge. The selection of appropriate actions in response to the situation is called tactics selection. Both stages require sophisticated reasoning and are closely linked. Having determined his current situation, a pilot needs to select and execute certain actions; these actions, in part, determine the way in which the situation evolves.

A pilot's reasoning process can be characterized as consisting of *beliefs* about the real world, *desires* that need to be satisfied, mental *plans* or procedures that satisfy certain desires, and committed partial plans or *intentions* that a pilot has adopted in response to external events or internal desires.

In obtaining situation awareness, a pilot must infer the beliefs, desires and intentions of other pilots from the behaviour of their aircraft. In tactics selection, a pilot must react to his beliefs about the current situation or advance his own desires.

While the problems of situation awareness and tactics selection are difficult for a single pilot in combat with a single enemy, the problems become far more complex when a team of pilots is in combat with an enemy team. The team as a whole needs to assess the situation by inferring not only the individual beliefs, desires, and intentions of individual pilots, but also the mutual beliefs, joint desires, and joint intentions of the entire enemy team. Similarly, tactics selection by a team is more difficult than the selection of tactics by a single pilot, because of the coordination and synchronization required.

[1]Human-in-the-loop simulation may not require modelling of pilot knowledge.

Elsewhere [13], we have discussed the selection of tactics by teams of pilots. In this paper, we concentrate on inferring the mental-states of other agents. We restrict our attention to multiple agents recognizing the mental-states of other agents. The recognition of joint mental attitudes [18] is beyond the scope of this paper.

## 3 Mental-State Recognition

In this section we illustrate informally our approach to the processes of plan execution and plan recognition using a sample scenario in air-combat modelling. Figures 7 and 8 show a number of plans, at different levels of granularity, to perform different types of intercepts. The syntax of the plans is based on our earlier work [6, 10] and is given in Figure 1.

| | | |
|---|---|---|
| ⟨plan⟩ | ::= | ⟨name⟩ ⟨invocation⟩ ⟨precond⟩ |
| | | ⟨postcond⟩ ⟨body⟩ |
| ⟨name⟩ | ::= | string |
| ⟨invocation⟩ | ::= | (DESIRE ⟨mode⟩ ⟨agent⟩ ⟨pln-expr⟩) |
| ⟨pln-expr⟩ | ::= | (! ⟨agent⟩ $\alpha$) |
| ⟨precond⟩ | ::= | $\alpha$ |
| ⟨postcond⟩ | ::= | $\alpha$ |
| ⟨body⟩ | ::= | ⟨node⟩ $\{-$ (⟨label⟩) $\rightarrow\}^+$ ⟨body⟩ \| |
| | | ⟨node⟩ $\{+$ (⟨label⟩) $\rightarrow\}^+$ ⟨body⟩ \| |
| | | ⟨node⟩ |
| ⟨label⟩ | ::= | (⟨mode⟩ ⟨agent⟩ $e$) \| (⟨mode⟩ ⟨agent⟩ ⟨pln-expr⟩) |
| ⟨mode⟩ | ::= | exec \| recog |
| ⟨node⟩ | ::= | symbol |
| $\alpha$ | ::= | well-formed propositional modal formula |

$\{b\}^+$ stands for one or more b's, where b is any terminal or non-terminal symbol.

Figure 1: BNF syntax for Plans

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 graph with the edges labelled with certain plan expressions. In the BNF syntax an OR-node is represented as $-$(⟨label⟩)$\rightarrow$ and an AND-node as $+$(⟨label⟩)$\rightarrow$ [2]. For a given proposition $\alpha$, the expression (exec $a$ (! $a$ $\alpha$)) means $a$ executes the *achievement* by $a$, of a state of the world where $\alpha$ is true. When the executing agent and the agent achieving the state of the world are the same, we simplify the notation and write it as (! $a$ $\alpha$). The expression (recog $a$ (! $b$ $\alpha$)) means $a$ recognizes the *achievement* by $b$ of a state of the world where $\alpha$ is true. The expression (exec $a$ $e$) and (recog $a$ $e$) mean the execution of the primitive plan and observation of the primitive plan $e$ by agent $a$ , respectively.

Now we introduce an example scenario from the air-combat modelling domain. In most typical air-combat scenarios aircraft are organized into pairs, with one aircraft playing the

[2] 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.

*lead* roles and the other, supporting, aircraft playing the *wing* roles. Pairs of such aircraft are then grouped together to form larger teams. Although the tasks to be done by the lead and wing aircraft are well specified for different phases of air-combat, these tasks are highly dependent on the situation awareness of the pilots at any given point in time. The tactics for carrying out these tasks can be represented as plans.

In this paper, we concentrate on the *intercept* phase of air-combat. A subset of the plans for intercepting enemy aircraft is given in Figure 7. We consider a 2 aircraft v. 2 aircraft scenario (called 2v2 scenario) with the agent *blue1* and *red1* playing the lead role and *blue2* and *red2* playing the wing role. The agents *blue1* and *blue2* act together as a team and the agents *red1* and *red2* act as an opposing team. The plan to intercept an enemy team is invoked if the lead agent *blue1* has the desire to intercept the enemy team (given as the invocation condition) and also believes that the enemy team is in formation (precondition). The plan to intercept involves the *blue1* agent doing either a *cut-off intercept* or a *pincer intercept.* Cut-off intercept involves both the lead and the wing aircraft staying together, obtaining an offset to the left or right of the enemy aircraft, and then attacking the enemy aircraft at the appropriate attacking range. In the case of pincer intercept the lead and wing separate from each other, obtain an offset on either side of the enemy aircraft and then attack. The plan for performing a cut-off intercept from the left is given by the plan *Cut-off Intercept Left.* The parallel tasks to be performed by *blue1* and *blue2* are given by the otgoing-arcs of an AND-node. The plans for *Cut-off Intercept Right* and *Pincer Intercept* are very similar.

Now consider the BDI-model of reactive planning. The *blue1* agent believes that the enemy team consisting of *red1* and *red2* is in formation. Furthermore, it acquires a desire to intercept the enemy team (this desire could be part of its overall desire to survive and/or accomplish the mission). This will result in the plan *Intercept* being adopted as an *intention* by the *blue1* agent. *Blue1* now acquires the desire to perform either a cut-off intercept or a pincer intercept. In the normal course of events, the agent will adopt one of these intercepts and carry it out. However, if there were some other important tasks to perform, e.g., the *blue1* agent notices that there are four other enemy aircraft approaching from the right and left, the agent may decide to *evade* enemy aircraft, rather than carry out its previous commitment to intercept the enemy team. This lends the plan execution process its reactive capability.

In reactive plan execution and 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, one after the other, until one of them succeeds, the observing agent should attempt to recognize all the applicable plans simultaneously. Otherwise, both the executing and observing agents are performing identical operations. The correspondence between execution and recognition[3] and the conditions under which they succeed are shown in Table 1. In this table, $p_1, \ldots p_n$ refer to the plans that can achieve $\alpha$; and $l_1, \ldots, l_n$ refer to the labels appearing on the outgoing edges of an OR-node or AND-node.

Now with this operational semantics let us run our example. We assume that both *blue1* and *red1* have the intercept plans in the plan library. In addition, the *blue1* agent has the recognition plans shown in Figure 8. The plan *Recognize Pincer Intercept Left* is invoked when the agent *blue1* has the desire to recognize the achievement of pincer intercept by *red1* and believes that the enemy team is in formation. To recognize such a pincer intercept the

[3]Note that in the case of $(!a\ \alpha)$ and OR-nodes it is sufficient for the execution to proceed sequentially; we have assumed that they are run as parallel processes mainly for convenience. However, all the recognitions have to be run in parallel.

| **Plan Entity** | **Execution** | **Recognition** | **Success Condition** |
|---|---|---|---|
| $(a\ e)$ | execute | observe | if $e$ succeeds for $a$ |
| $(!\ a\ \phi)$ with $p_1 \ldots p_n$ | in parallel run $p_1$ to $p_n$ | in parallel run $p_1$ to $p_n$ | if one of $p_i$ succeeds |
| OR-node with $l_1 \ldots l_n$ | in parallel run $l_1$ to $l_n$ | in parallel run $l_1$ to $l_n$ | if one of $l_i$ succeeds |
| AND-node with $l_1 \ldots l_n$ | in parallel run $l_1$ to $l_n$ | in parallel run $l_1$ to $l_n$ | if all of $l_i$ succeeds |

Table 1: Comparison of Execution and Recognition

agent *blue1* has to recognize *red1* doing a cut-off manouvre to the left and *red2* doing a cut-off manouvre to the right. This in turn leads to other plans for recognizing cut-off manouvres.

Assume that *red1* is doing a pincer intercept. The executing agent can fulfill its desire by adopting a plan to intercept from the left or from the right. Before *blue1* can recognize this desire it has to be *attentive* to this desire. In other words, *blue1* should decide to recognize the execution of a pincer intercept. At this particular phase of combat (i.e., intercept phase) it is reasonable for the *blue1* agent to be attentive towards all intercept desires. The approach to reactive recognition presented in this paper assumes that in many domains, given a particular situation, the observing agent will have a reasonable knowledge of the set of desires that the executing agent is likely to pursue and may decide to be attentive to these desires.

Now, given that *blue1* is attentive towards a pincer intercept it has to adopt the plans to recognize both the pincer intercept from the left and the right. This in turn would result in *blue1* adopting the desires to recognize the cut-off manouvres from the left by *red1* and from the right by *red2*. Subsequently this would result in *blue1* adopting the desires to recognize *red1* reaching roll-range to the left and *red2* reaching roll-range to the right. Let the agent *red1* execute a pincer intercept, reaching roll-range and subsequently sort-range on the left and simultaneously *red2* reaching roll-range and subsequently sort-range on the right. Given that the agent *blue1* has desires and intentions to recognize these events the agent *blue1* will conclude that *red1* is attempting to perform a pincer intercept to the left.

So far we have described the recognition process of agent *blue1*. While *blue1* is performing the recognition, it can also be executing other tasks, e.g., operating its radar to observe other enemy aircraft. Also, if there is a demand on its computational resources and there are other higher priority tasks e.g., evading other aircraft, it can suspend or abort its current recognition tasks.

The full expressive power of the approach can be exploited by combining the reactive recognition and execution processes. For example, to perform intercepts the agent *blue1* can either adopt the plan *Intercept* (as shown in Figure 7) or the more complex plan *Intercept (Complex)*. In the latter plan, *blue1* observes what the enemy team is doing and acts accordingly. First, it tries to recognize if the enemy team is performing a pincer intercept or a cut-off intercept. This leads to the recognition process as described above. Once *red1* and *red2* are in sort-range the agent *blue1* has recognized that it is the desire of *red1* to perform a pincer intercept. This is in turn results in *blue1* believing that *red1* has the desire to perform a pincer intercept. Based on this *blue1* performs a cut-off intercept to the left (i.e., *Cut-Off Intercept Left (Complex)*). However, if *red1* had performed a cut-off intercept, the *blue1* agent would have performed a pincer intercept.

Having discussed the process informally through an air-combat example, we shall now provide the algorithms for integrated reactive plan execution and recognition embedded within the mental-state of an agent.

## 4 Algorithms

The mental-state interpretation of means-end plan execution is well known within the community [4, 17, 20]. One can provide an analogous mental-state interpretation of means-end 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.

An abstract BDI-interpreter for performing reactive plan execution within the context of the beliefs, desires (or goals), intentions, and plans of an agent was given elsewhere [17]. In this paper, we extend this BDI-interpreter to reactive recognition and integrate both reactive plan execution and recognition.

Before we go into the details of the BDI interpreter we describe some of the data structures used by the BDI interpreter.

**Trigger:** A trigger is an internal event which consists of a *trigger type* and a *trigger value.* The interpreter responds to triggers by altering its internal state.

⟨trigger⟩ := ⟨type⟩ ⟨value⟩

**Trigger Type:** The type of a trigger can be any one of the following: BEL-ADD, BEL-REM, DES-ADD, DES-REM, INT-ADD, INT-REM, ACT-SUC, PLN-SUC, ACT-FAL, PLN-FAL. The first six types were discussed elsewhere [17] and they initiate the addition or removal of beliefs, desires, and intentions, respectively. The last four events are genereated when a primitive act or plan succeeds or fails, respectively.

⟨type⟩ := BEL-ADD ⟨bel-value⟩ | BEL-REM ⟨bel-value⟩ |
DES-ADD ⟨des-value⟩ | DES-REM ⟨des-value⟩ |
INT-ADD ⟨int-value⟩ | INT-REM ⟨int-value⟩ |
ACT-SUC ⟨act-res-val⟩ | ACT-FAL ⟨act-res-val⟩ |
DES-SUC ⟨des-res-val⟩ | DES-FAL ⟨des-res-val⟩ |
PLN-SUC ⟨pln-res-val⟩ | PLN-FAL ⟨pln-res-val⟩

**Trigger Value:** The value of a trigger can be any one of the following: (a) belief value which consists of a proposition; (b) desire value which consists of the *mode*, an agent, and a plan expression; (c) intention value which consists of the *end* that is to be intended as part of the *means* in a particular *intention tree* or the means that is to be intended as part of satisfying an end in a particular intention tree. (d) plan value which consists of a *means*, (towards a certain) *end*, and an *intention tree.*

⟨value⟩ := ⟨bel-value⟩ | ⟨des-value⟩ | ⟨int-value⟩ |
⟨act-res-val⟩ | ⟨des-res-val⟩ | ⟨pln-res-val⟩

⟨bel-value⟩ := ⟨prop⟩

⟨des-value⟩ := ⟨mode⟩ ⟨agent⟩ ⟨pln-expr⟩

⟨act-res-val⟩, ⟨des-res-val⟩ := ⟨end⟩ ⟨means⟩ ⟨intention-tree⟩

⟨pln-res-val⟩ := ⟨means⟩ ⟨end⟩ ⟨intention-tree⟩

⟨int-value⟩ := ⟨end⟩ ⟨means⟩ ⟨intention-tree⟩ | ⟨means⟩ ⟨end⟩ ⟨intention-tree⟩

**Mode:** The mode can either be an execution mode (denoted by the keyword "exec") or a recognition mode (denoted by the keyword "recog").

⟨mode⟩ := exec | recog

**Option:** An option is a trigger followed by either a plan-name or an action.

⟨option⟩ := ⟨trigger⟩ ⟨plan-name⟩

**End:** An end is either a desire to achieve/recognize a proposition or to execute/observe a primitive action. This is denoted by either the keyword DESIRE followed by a desire value or the keyword PRM-ACT followed by an action value.

⟨end⟩ := DESIRE ⟨des-value⟩ | PRM-ACT ⟨prm-act-value⟩

**Means:** The means is given by the plan which contains the keyword PLAN followed by the mode, name, invocation, context, effects, and current-node.

⟨means⟩ := PLAN ⟨mode⟩ ⟨name⟩ ⟨invocation⟩ ⟨context⟩ ⟨effects⟩ ⟨current-node⟩

**End Tree:** An end tree consists of an end followed by a list of zero or more means trees. Each means tree is a partial unwinding of a means to satisfy the end.

⟨end-tree⟩ := ⟨end⟩→⟨means-tree⟩*

**Means Tree:** A means tree consists of a means followed by a list of one or more end trees. Each end tree is a partial unwinding of the end that needs to be satisfied for the program control to move to the next node.

⟨means-tree⟩ := ⟨means⟩→ ⟨end-tree⟩+

**Intention Tree:** An intention tree is a special type of end tree with at least one means tree.

⟨intention-tree⟩ := ⟨end⟩→ ⟨means-tree⟩+

The procedure **reactive-reasoner** (see Figure 2) is called in each cycle with the set of plans P, the current mental state $M_S$, and a list of triggers that contain both internal and external events. The result of running the reactive reasoner is given by the mental state $M_T$. The agent first generates all options for all the triggers in the trigger list, i.e., the plans that can satisfy a given trigger. The agent then deliberates on these options and selects a subset of these options to which it commits. The mental state of the agent is then updated with respect to this set of selected options. The agent then runs the updated intention structure resulting in a new mental state.

We take the mental state to be the set of beliefs and intentions. Note that we have not taken the desires to be part of the mental state. This is because desires can be of two types: *intrinsic desires* that arise from within the agent and *secondary desires* that arise as a result of the agent attempting to satisfy its intrinsic desires. Once the agent chooses to satisfy an intrinsic desire it becomes an intention and all the secondary desires are represented as intentions. An intrinsic desire that is not chosen by the agent is dropped and is not maintained from one time point to the other. However, the beliefs and intentions of the agent are maintained from one time point to the next. This implementation is a reflection of the different axioms of commitment we have adopted for desires, as compared to beliefs and intentions [17].

```
procedure reactive-reasoner (P, $M_S$, trigger-list, $M_T$)
options := option-generator(P, $M_S$, trigger-list);
selected-options := deliberate($M_S$, options);
M := update-mental-state($M_S$, selected-options);
$M_T$ := run-intention-structure(M).
```

Figure 2: Reactive Reasoner procedure

The procedure **option-generator** is straightforward and generates the set of applicable plans (as shown in Section 4) for each trigger. The **deliberate** procedure could include domain-independent decision-theoretic analysis for choosing options with the highest expected utility or domain-dependent heuristics for filtering the options. Space precludes us from discussing the **deliberate** procedure further.

An important part of the **reactive-reasoner** is the procedure that updates the mental-state of the agent by processing the options selected by the deliberate procedure. This involves a case-by-case analysis of all the events and how they change the mental-state of the agent. The details of the procedure **update-mental-state** is given in Figure 3.

The procedure **update-mental-state** processes updates (such as belief, desire, and intention, additions and removals) and results (such as the success or failure of actions, desires, and plans). These are given by the procedures **process-updates** and **process-results**. The current mental state of the agent is then returned by the **update-mental-state** procedure.

```
procedure update-mental-state($M_S$, selected-options)
Get the selected triggers from the selected-options;
For each selected-trigger in selected-triggers do
    Case type of selected-trigger is
        BEL-ADD, BEL-REM, DES-ADD, DES-REM, INT-ADD, INT-REM:
            process-updates($M_S$, selected-options);
        ACT-SUC, ACT-FAL, DES-SUC, DES-FAL, PLN-SUC, PLN-FAL:
            process-results($M_S$, selected-options);
Return the updated belief and intention states.
```

Figure 3: Procedure for updating mental-state

Belief add and remove events change the set of beliefs of the agent. These changes are assumed to be internally consistent and therefore we avoid the problem of belief revision [1].

Adding a desire results in the agent creating a new intention tree with the desire as the end and the plans that can satisfy that end as the means of the intention tree. Removing a desire (which should be an end at the root of an intention tree) results in the entire intention tree being removed from the intention structure.

Adding an intention results in the agent modifying the intention tree with an end tree, i.e., the end that needs to be added and the means as the children that can achieve that end. Removing an intention (which in this case can either be an end or a means) results in the end or means (given by the first argument) being removed from the intention tree. The addition and removal of beliefs, desires, and intentions is given by the procedure **process-updates** in Figure 4.

```
procedure process-updates($M_S$, selected-options)
Get the selected triggers from the selected-options;
For each selected-trigger in selected-triggers do
    Case type of selected-trigger is
        BEL-ADD, BEL-REM:
            Update $B_S$ with the proposition given in the selected-trigger
        DES-ADD:
            Create a new intention tree as follows:
                (a) The root of the intention tree is the value of the selected trigger
                (b) The children of the root are the various plans that appear in
                the selected options for the selected trigger
            Add the new intention tree to $I_S$
        DES-REM:
            Remove the intention tree whose root node matches the value of the
            selected trigger from $I_S$.
        INT-ADD:
            Add the end as the child for the means in the intention tree.
            Add the plans that appear in the selected options as the children for the end.
        INT-REM:
            Remove the end (means) as the child for the means (end) in the intention tree.
Return the updated belief and intention states.
```

Figure 4: Procedure for processing updates

When a primitive action or desire is successful, either in the execution or recognition modes, the end corresponding to that action or desire is removed from the intention tree. If the parent of this end, i.e., means or plan, is currently at an OR node we can remove all the other edges and proceed to the next node, because semantically it is sufficient for one edge to succeed for the OR node to succeed. In the case of an AND node one can proceed to the next node, only if the end that succeeded was the last edge of the node. If one proceeds to the next node and the next node is an END node then the entire plan is successful and a successful plan event is posted.

When a primitive action or desire fails in the execution mode, the end corresponding to that action or desire is removed from the intention tree. If the parent of this end, i.e., means or plan, is currently at an AND node we can get rid of all the other edges and fail the plan, because semantically an AND node fails even if one of its edges fails. In the case of an OR node one fails the plan, only if the end that succeeded was the last edge of the node.

Note that we have not said anything in the case of a primitive action or desire failing in the recognition mode. In other words, the agent is committed to observing/recognizing the primitive actions and desires and will not fail the plan until it has recognized them. In essence, the intentions are in a suspended state waiting for the successful observation of primitive actions to take place. To avoid *blind commitment* towards recognition one can have a timeout on how long the agent waits before abandoning its desires.

In the case of a successful plan event the agent adds belief add events corresponding to the effects of the plan, sends an event to remove the intention corresponding to the plan, i.e., means, and sends a successful desire event corresponding to the desire, i.e., end. For plan failure events the effects are not added and the agent sends appropriate intention remove and desire failure events. Processing the success or failure of actions, desires, and plans is given by the procedure **process-results** in Figure 5.

At the end of the **update-mental-state** procedure we could have a number of intention trees in the intention structure whose leaves are means (i.e., plans) or ends (i.e., desires or primitive actions). The procedure **run-intention-structure** runs each intention tree in this intention structure (see Figure 6).

If the leaf node is a primitive action then the agent either executes the action or observes the action. As discussed earlier, in the case of execution the agent performs the action once and the success or failure of the act is informed to the agent by the environment as ACT-SUC or ACT-FAL events. In the case of recognition if the agent cannot observe the successful performance of the action, it waits indefinitely (or until the intention is timed out).

If the leaf node is a plan the agent executes the plan by adding new INT-ADD events for each outgoing edge of the current node. For example, an AND node which has multiple outgoing edges will have multiple INT-ADD events being generated. These events will then be processed in the next cycle by the **update-mental-state** procedure.

A leaf node which happens to be a desire, i.e., DESIRE ⟨des-value⟩ is not processed by the **run-intention-structure** because the INT-ADD events for such cases would have been generated by the **update-mental-state** procedure.

## 5 Air-Combat Scenario

Now let us go back to the example considered earlier. The agent *blue1*'s desire is to execute an intercept. This corresponds to the following trigger in the trigger list: DES-ADD EXEC blue1 (! blue1 (intercept)). The **reactive-reasoner** procedure first produces the various plans that can respond to this event, namely Intercept (Complex). Assuming that the **deliberator** selects these options the **update-mental-state** procedure will process the DES-ADD trigger and create an intention tree whose root node is the triggering desire (see E1 of Figure 9) and whose child is the plan that can satisfy the desire (i.e., M1 node of Figure 9[4]). At this stage the leaf of the intention tree is M1 which is a plan. The **run-intention-structure** results in the following events being sent as triggers: INT-ADD (DESIRE RECOG blue1 (! red1 (cut-off-intercept))) M1 INT-TREE and INT-ADD (DESIRE RECOG blue1 (! red1 (pincer-intercept))) M1 INT-TREE. This completes one cycle of the BDI interpreter.

In the next cycle there will be at least two triggers in the trigger list, i.e., the INT-ADD events discussed above. In addition, the trigger list may contain other external events.

[4]We have not reproduced all the different fields of E1, M1, and all the other nodes due to lack of space. For example, the agent argument is omitted and "exec" and "recog" are abbreviated by "E" and "R", respectively.

Options will be generated for all the triggers in the trigger list. The plans *Recog. Cut-Off Intercept Left*, *Recognize Cut-Off Intercept Right* will respond to the first trigger and the plans *Recognize Pincer Intercept Left* and *Recognize Pincer Intercept Right* will respond to the second trigger. The deliberator then selects a subset of these options according to some pre-defined criteria. For example, if the deliberator selects options with the highest priority, then depending on whether the external events have a higher priority or not the deliberator procedure will choose the two INT-ADD events. Assume that there are no other higher priority options. The two INT-ADD events will be processed by the update-mental-state procedure. This will result in the intention tree having M2, M3, M4, and M5 as its leaves. The run-intention-structure will result in eight additional INT-ADD events being generated. The process continues for a couple of cycles before we arrive at the means-end tree shown in Figures 8 and 9 that contain all primitive plans or actions as their leaves.

If none of these actions is observed immediately the intention is suspended until at least one of these acts is observed. As the red team is doing a pincer intercept to the left, after some time, *blue1* will be able to observe the primitive acts of *red1* getting within roll-range on the left and *red2* getting within roll-range on the right. This will result in the following two ACT-SUC events being generated: ACT-SUC (PRM-ACT RECOG blue1 (in-roll-range-on-left red1)) M10 INT-TREE and ACT-SUC (PRM-ACT RECOG blue1 (in-roll-range-on-right red2)) M11 INT-TREE.

In the next cycle both the ACT-SUC events will be processed by update-mental-state procedure. The node E16 and E17 will be removed from the intention tree and the current nodes of the plans corresponding to M10 and M11 will be updated. This will result in intentions to recognize the primitive plans of *red1* being in sort-range on the left and *red2* being in sort-range on the right. Once these primitive plans are observed, the following PLN-SUC events will be generated as follows: PLN-SUC M10 E8 INT-TREE and PLN-SUC M11 E9 INT-TREE.

Processing the PLN-SUC event will result in a DES-SUC event. Processing this will result in M10, E8 and M11, E9 being dropped as the desires have been achieved. After this there will be a PLN-SUC event for M4. Once this happens the means-end tree with M5 as the root node can be aborted and the desire E3 succeeds. This results in E2 being removed as E2 and E3 are parts of an OR-node. Note that at this stage *blue1* believes that *red1* has the desire to perform a pincer intercept from the left. Subsequently, the agent can then choose an appropriate counter-tactic. In this case, it will result in a desire to choose an intercept, which will result in *blue1* adopting the plan to perform a cut-off intercept to the left. This process of incrementally updating the intention-tree will continue till the top-level deisres are satisfied.

There are two primary differences between the BDI interpreter given above and the Procedural Reasoning System [6]. First, we have presented a propositional variant of the PRS interpreter which is a restricted first-order system with full unification. Second, PRS is a reactive planning system that expands the means-end tree in a depth-first manner for efficiency reasons. As a result the interpreter makes use of a means-end stack to represent a top-level intention and all its children. Therefore in PRS, Reactive recognition has to be explicitly programmed by the user.

Elsewhere [15, 16] we have described a family of modal temporal logics with modal operators for beliefs, desires, and intentions and a branching-time temporal structure based on CTL* [5]. These logics describe the static and dynamic relationships amongst the mental attitudes of agents. We have introduced a *dynamic agent logic* for a limited form recognition called means-end recognition elsewhere [14]. Combining these so called BDI-logics with the dynamic agent logic provides a powerful logic to examine the behaviours of resource-bounded agents capable of planning based on their current mental state and recognizing the mental state of other agents.

Although the initial motivation for developing a theory of reactive recognition was the need to infer the mental states of opponents in an air-combat modelling system, the approach may also be able to be used fruitfully in a number of collaborative domains such as discourse understanding [11] and automated tutoring systems.

## 6 Comparison and Conclusions

**Plan Recognition:** Early work by Allen and Perrault [2] and more recently by Litman and Allen [11] 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 is 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.

Kautz's approach proceeds bottom-up, creating an explanation for each observation and then merging these explanations, while the *reactive 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 reactive recognition algorithm with a trigger such as (recog blue1 (!red1 (pincer-intercept))) before the intercept has started; otherwise the agent will not be able to recognize the intercepts, even if it had plans to recognize intercepts. This is not the case in Kautz's approach.

**Situation Awareness:** The simplest approach to situation awareness is to assume that all pilots know the positions and types of all other aircraft present, and have the means of measuring their velocities and observing their actions, but have no means of reasoning about the opponent's possible plans. That is, the pilots simply react to the opponent's observed actions in trying to achieve their own goals. This approach, which minimises the representation of situation awareness, may be adequate for comparing the relative effectiveness of different tactics against a given enemy tactic, but does not provide an adequate representation of tactical decision making by the pilots and does not take account of pre-mission briefing of pilots.

Azarewicz *et. al.* [3] have developed a system which is based on recognizing the an opponent's plans from a given set of possible plans, based on observing manoeuvres and actions. In this paper we have extended this approach and developed a formalism for recognising beliefs, desires and intentions of other agents in the environment. This allows our existing PRS-based model [13] to represent that important part of situation awareness which involves "thinking like the enemy". Computation time is constrained by having a limited number of possible plans to be considered.

REPLAI (REcognition of PLans And Intentions) [19] recognizes plans and intentions of agents in the domain of soccer games. It makes use of a plan hierarchy that captures the specialization hierarchy (e.g., a pincer intercept is a kind of intercept) and the decomposition hierarchy (e.g., a "move" plan can be decomposed into "pick" and "put" primitive plans or actions in the blocks world). The plan hierarchy is split into two distinct parts, with the higher-level capturing the specialization and the lower-level capturing the decomposition hierarchies. Recognition proceeds top-down from the specialization hierarchy and bottom-up from the decomposition hierarachy.

There are a lot of similarities between REPLAI and the approach presented in this paper. Both use libraries of plans and proceed top-down as well as bottom-up. However, due to the separation of hierarchies in REPLAI it is not possible to have decomposition of goals (or desires) at the higher levels. Also, the recognition process is not embedded within the beliefs of the agents and their overall planning tasks. As a result, one cannot have preconditions to plans that are complex mental attitudes (such as beliefs or desires of other agents) and then execute plans based on such preconditions.

In summary, the primary contribution of this paper is in presenting an integrated approach to reactive plan execution and plan recognition that is applicable to a broad class of problems where the environment is changing rapidly and the agents are resource-bounded. By embedding these processes within a framework of the mental-state of agents, they also facilitate mental-state recognition. The novelty of the approach is in modelling the "attentiveness" of agents to constrain the execution and recognition processes. The approach looks promising for assessing the situation awareness of pilots in air-to-air combat.

**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

[1] C. Alchourron, P. Gardenfors and D. Makinson. On the logic of theory change: partial meet contraction functions and their associated revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] J. F. Allen, and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, 15:143–178, 1980.

[3] J. Azarewicz, G. Fala, R. Fink, and C. Heitheker. Plan recognition for airborne tactical decision making. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, pages 805–811, 1986.

[4] M. E. Bratman, D. Israel, and M. E. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355, 1988.

[5] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen editor, *Handbook of Theoretical Computer Science: Volume B, Formal Models and Semantics.* Elsevier Science Publishers and MIT Press, Amsterdam and Cambridge, MA, pages 995–1072, 1990.

[6] M. P. Georgeff and A. L. Lansky. Procedural knowledge. In *Proceedings of the IEEE Special Issue on Knowledge Representation*, volume 74, pages 1383–1398, 1986.

[7] M.P. Georgeff. *Planning.* Annual Reviews, Inc., Palo Alto, California, 1987.

[8] S. Goss, editor. *Proceedings of the IJCAI-91 Workshop on Situation Awareness.* International Joint Conference on Artificial Intelligence, Sydney, Australia, 1991.

[9] H. Kautz. A circumscriptive theory of plan recognition. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, MA, 1990.

[10] David Kinny, Magnus Ljungberg, Anand Rao, Elizabeth Sonenberg, Gil Tidhar, and Eric Werner. Planned team activity. In *Proceedings of the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW '92*, Viterbo, Italy, 1992.

[11] D. J. Litman and J. Allen. Discourse processing and commonsense plans. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, MA, 1990.

[12] M. E. Pollack. Plans as complex mental attitudes. In P. R. Cohen, J. Morgan, and M. E. Pollack, editors, *Intentions in Communication.* MIT Press, Cambridge, MA, 1990.

[13] A. Rao, D. Morley, M. Selvestrel, and G. Murray. Representation, selection, and execution of team tactics in air combat modelling. In *Proceedings of the Australian Joint Conference on Artificial Intelligence, AI'92*, 1992.

[14] A. S. Rao. Means-end plan recognition: Towards a theory of reactive recognition. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KRR-94)*, Bonn, Germany, 1994.

[15] A. S. Rao and M. P. Georgeff. Asymmetry thesis and side-effect problems in linear time and branching time intention logics. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, Sydney, Australia, 1991.

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

[17] A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning.* Morgan Kaufmann Publishers, San Mateo, CA, 1992.

[18] A. S. Rao, M. P. Georgeff, and E. A. Sonenberg. Social plans: A preliminary report. In E. Werner and Y. Demazeau, editors, *Decentralized A.I. – Proceedings of the Third European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds*, Amsterdam, The Netherlands, 1992. Elsevier Science Publishers.

[19] Gudula Retz-Schmidt. A replai of soccer: Recognizing intentions in the domain of soccer games. In *Proceedings of the Eighth European Conference on Artificial Intelligence (ECAI-88)*, pages 455–457, 1988.

[20] Y. Shoham. Agent0: A simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, pages 704–709, 1991.

```
procedure process-results($M_S$, selected-options)
Get the selected triggers from the selected-options;
For each selected-trigger in selected-triggers do
    Case type of selected-trigger is
        ACT-SUC, DES-SUC:
            Remove the end given by the first argument of the selected trigger and all its children.
            Go to the parent of end which will necessarily be a means.
            Case current node of means is
                OR: Remove all the children of means from the intention tree
                    Make the current node point to the next node in the plan
                AND: If there are no more children of means then make the current
                    node point to the next node in the plan
            If the current node is an END node then
                send PLN-SUC event with means, parent of means, and the intention
                structure as its three arguments
        ACT-FAL, DES-FAL:
            If mode of the selected trigger is EXEC then
                Remove the end given by the first argument of the selected trigger.
                Go to the parent of end which will necessarily be a means.
                    Case the current node of means is
                        AND: Remove all the children of means from the intention tree
                        send PLN-FAL event with means, parent of means, and the intention
                        structure as its three arguments
                        OR: If there are no more children of means then
                        send PLN-FAL event with means, parent of means, and the intention
                        structure as its three arguments
        PLN-SUC:
            send BEL-ADD events for all the effects of the plan
            send INT-REM event with the same three arguments as the PLN-SUC event
            send DES-SUC event with the second argument (end), parent of the second
            argument (means), and the thrid argument (intention-structure) of the PLN-SUC event
        PLN-FAL:
            send INT-REM event with the same three arguments as the PLN-FAL event
            send DES-FAL event with the second argument (end), parent of the second
            argument (means), and the thrid argument (intention-structure) of the PLN-FAL event
Return the updated belief and intention states.
```

Figure 5: Procedure for processing results

```
**procedure** run-intention-structure(Intention-structure)
for int-tree in intention-structure
    for leaf in leaves-of(int-tree)
        Type of leaf is
        PRM-ACT:
            If the mode of the leaf is EXEC then execute(leaf) else observe(leaf);
        PLAN:
            for edge in edges-of(current-node-of(leaf))
                send INT-ADD event with the label of edge, leaf, and int-tree as its arguments.
```

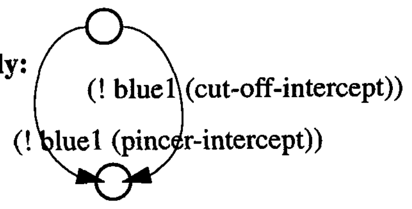
Figure 6: Procedure for running the intention structure

**PLAN:** *Intercept*

**invocation condition:**
(DESIRE
(! blue1 (intercept)))

**precondition:**
(BELIEF blue1
(in-formation (red1 red2)))

**body:**

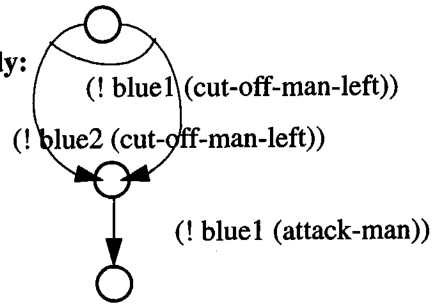


**PLAN:** *Cut-Off Intercept Left*

**invocation condition:**
(DESIRE
(! blue1 (cut-off-intercept)))

**precondition:**
(BELIEF blue1
(in-formation (red1 red2)))

**body:**

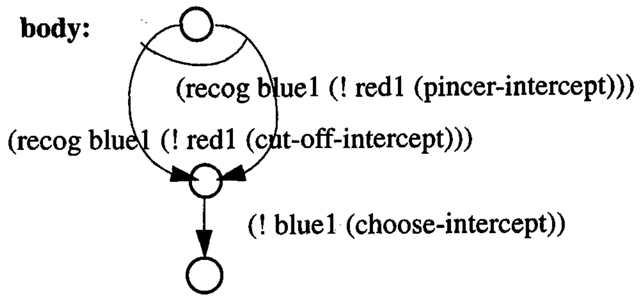


**PLAN:** *Intercept (Complex)*

**invocation condition:**
(DESIRE
(! blue1 (intercept)))

**precondition:**
(BELIEF blue1
(in-formation (red1 red2)))

**body:**


(recog blue1 (! red1 (pincer-intercept)))
(recog blue1 (! red1 (cut-off-intercept)))
(! blue1 (choose-intercept))

**PLAN:** *Cut-off Intercept Left (Complex)*

**invocation condition:**
(DESIRE
(! blue1 choose-(intercept)))

**precondition:**
(AND
(BELIEF blue1
(in-formation (red1 red2)))
(BELIEF blue1
(DESIRE (! red1 (pincer-intercept)))

**body:**

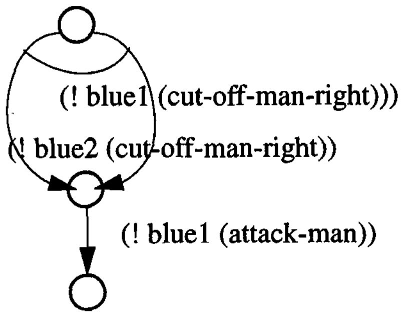


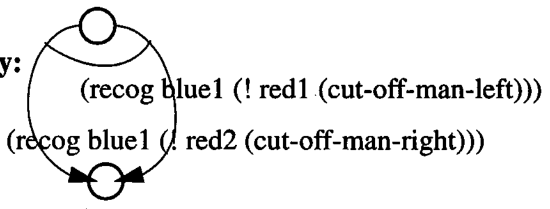
Figure 7: Plans for executing intercepts

**PLAN:** *Recog.Pincer Intercept Left*

**invocation condition:**
(DESIRE recog blue1
(! red1 (pincer-intercept))))

**precondition:**
(BELIEF blue1
(in-formation (red1 red2)))

**body:**

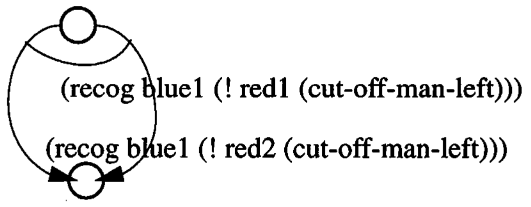

(recog blue1 (! red1 (cut-off-man-left)))
(recog blue1 (! red2 (cut-off-man-right)))

**PLAN:** *Recog. Cut-Off Intercept Left*

**invocation condition:**
(DESIRE recog blue1
(! red1(cut-off-intercept)))

**precondition:**
(BELIEF blue1
(in-formation (red1 red2)))

**body:**


(recog blue1 (! red1 (cut-off-man-left)))
(recog blue1 (! red2 (cut-off-man-left)))

**PLAN:** *Recognize Cut-Off Man.Left1*

**invocation condition:**
(DESIRE recog blue1
(! red1 (cut-off-man-left )))

**body:**

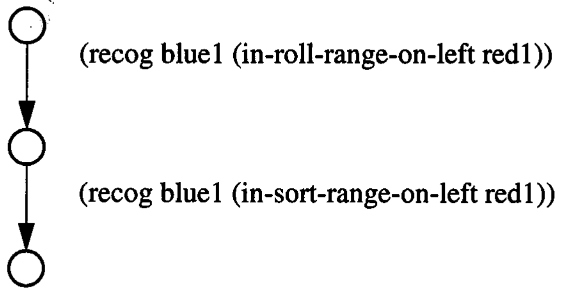

(recog blue1 (in-roll-range-on-left red1))
(recog blue1 (in-sort-range-on-left red1))

Figure 8: Plans for Recognizing Intercepts

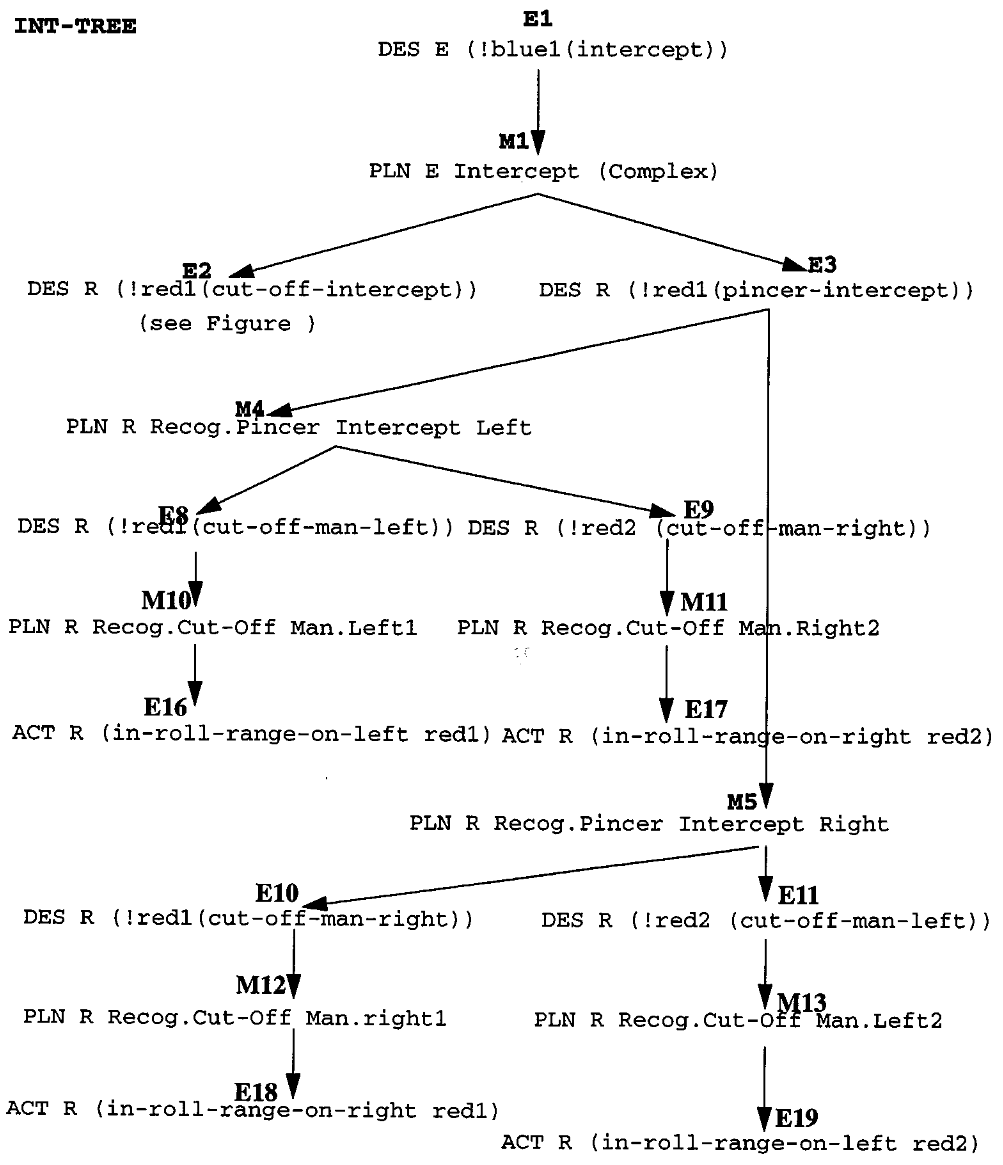

INT-TREE
E1
DES E (!blue1(intercept))
M1
PLN E Intercept (Complex)
E2
DES R (!red1(cut-off-intercept))
(see Figure )
E3
DES R (!red1(pincer-intercept))
M4
PLN R Recog.Pincer Intercept Left
E8
DES R (!red1(cut-off-man-left))
E9
DES R (!red2 (cut-off-man-right))
M10
PLN R Recog.Cut-Off Man.Left1
M11
PLN R Recog.Cut-Off Man.Right2
E16
ACT R (in-roll-range-on-left red1)
E17
ACT R (in-roll-range-on-right red2)
M5
PLN R Recog.Pincer Intercept Right
E10
DES R (!red1(cut-off-man-right))
E11
DES R (!red2 (cut-off-man-left))
M12
PLN R Recog.Cut-Off Man.right1
M13
PLN R Recog.Cut-Off Man.Left2
E18
ACT R (in-roll-range-on-right red1)
E19
ACT R (in-roll-range-on-left red2)

Figure 9: Means-End Tree for Execution of Intercepts

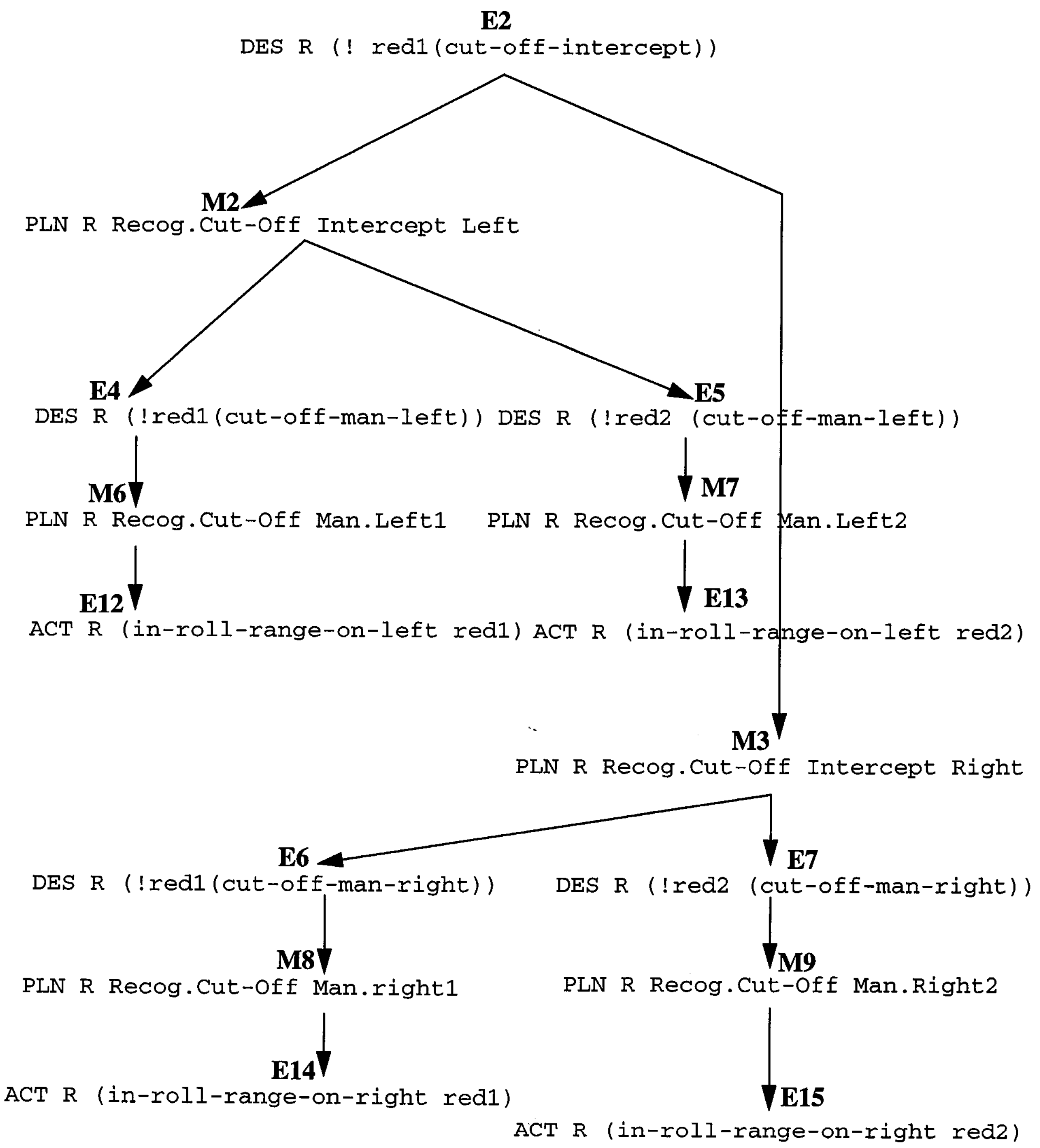

E2
DES R (! red1(cut-off-intercept))
M2
PLN R Recog.Cut-Off Intercept Left
E4
DES R (!red1(cut-off-man-left))
E5
DES R (!red2 (cut-off-man-left))
M6
PLN R Recog.Cut-Off Man.Left1
M7
PLN R Recog.Cut-Off Man.Left2
E12
ACT R (in-roll-range-on-left red1)
E13
ACT R (in-roll-range-on-left red2)
M3
PLN R Recog.Cut-Off Intercept Right
E6
DES R (!red1(cut-off-man-right))
E7
DES R (!red2 (cut-off-man-right))
M8
PLN R Recog.Cut-Off Man.right1
M9
PLN R Recog.Cut-Off Man.Right2
E14
ACT R (in-roll-range-on-right red1)
E15
ACT R (in-roll-range-on-right red2)

Figure 10: Means-End tree for Recognizing Intercepts