A Message Passing Algor n for Plan Recognition

Dekang Lin
Dept. of Advanced Computing & Engineering
Alberta Research Council
Calgary, Alberta, Canada T2E 7H7
dekang@noah.arc.ab.ca

Randy Goebel
Dept. of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
goebel@cs.Ualberta.ca

Abstract

We present a general framework for plan recognition whose formulation is motivated by a general purpose algorithm for effective abduction. The knowledge representation is a restricted form of first order logic, which is made computationally explicit as a graph structure in which plans are manifest as a special kind *of* graph walk. Intuitively, plans are fabricated *by* searching an action description graph for relevant connections amongst instances of observed actions.

The class of plans for which our method is applicable is wider than those previously proposed, as both recursive and optional plan components can be represented. Despite the increased generality, the proposed message-passing algorithm has an asymptotic upper bound that is an improvement on previous related work.

1 Introduction

In cases where we have incomplete specifications of how agents act in environments, the motivation for plan recognition is to somehow attribute a plan to the agent's observed actions, in order to help create a less incomplete specification of behaviour. The extra information, regardless of how it is fabricated, is typically used to do things like explain actions already taken, or to predict actions to be taken. Applications of plan recognition have been found in discourse understanding [Litman and Allen, 1991], intelligent interfaces [Goodman and Litman, 1990], cooperative problem solving [Lochbaum *et* al., 1990] and the analysis of goal-directed behavior in general.

Kautz [Kautz, 1987] proposed a formal theory of plan recognition- The advantages *of* Kautz's theory include its formal declarative semantics and expressive richness of its plan representation. Unfortunately, Kautz's plan recognition algorithm is exponential in the size of the knowledge base [Kautz, 1987, p. 119] and thus may prove difficult to scale up. Vilain [Vilain, 1990] showed that, by imposing certain restrictions on plan representation, the plan recognition problem may be turned into a context free grammar parsing problem, which has well-known polynomial solutions. However, in order for Vilain's scheme to work, the plan components (steps) must be fully ordered and cannot be shared or interleaved. These conditions appear to be too restrictive in most applications.

Rather than restrict the plan representation formalism to suit parsing algorithms, we propose a new algorithm for plan recognition that is a generalization of a message passing algorithm for context-free grammar parsing [Lin and Goebel, 1990]. The plans libraries are represented by first order logic (FOL) formulas, as in [Kautz, 1987]. The solutions for the plan recognition problem are scenarios which, together with background knowledge (plan libraries), logically entail the observations. The complexity of our algorithm is linear in size (number of axioms) of the knowledge base. If the same set of restrictive assumptions are made as in [Vilain, 1990], the complexity of our algorithm becomes $0(|H|n^3)$, where *[H]* is the size (number of nodes and links) of the plan hierarchy and *n* is the number of observations to be explained. In contrast, the complexity of Vilain's algorithm is $0(|H|^2n^3)$.

Another contribution of this research is a formal, declarative treatment of specificity of explanations. Specificity has always been a confusing issue in plan recognition. Some have argued that the most specific explanation should be preferred because more specific information should preempt the more general. Alternatively, a more general statement about a plan instance is more likely to be true and therefore preferable. In [Kautz, 1987], specificity preference is not formally specified. Rather, it is dealt with by a consider-spac flag in the procedural description of the plan recognition algorithm. In Vilain's simplification this issue is totally ignored. We have taken the approach where preference is given to the *most general* of the scenarios that consist only of arguments that are the *most specific* with respect *to* the observations.

2 Plan Representation

Our plan representation scheme is similar to that of [Kautz, 1987], but is more general in that recursive and optional plan components can be represented. On the other hand, the constraints in Kautz's plan representation language can be any FOL formula that does not

contain predicates representing plan types, here, the constraints have more restrictive patterns.

The motivation for the more restrictive patterns of constraints is a computational one. A major source of complexity in Kautz's algorithm is the fact that the explanation graphs in Kautz's algorithm are not able to contain local ambiguities. That is, two or more alternative explanations of a subset of observations from a plan type may combine with a similar set of alternatives for a different subset to produce an exponential growth in the number of global alternatives. However, local ambiguity may be globally irrelevant, in which case containment implies that local ambiguities are not multiplied. Our message passing algorithm is able to block the propagation of local ambiguities; a more restrictive form of constraints in our plan representation language guarantees that the global consistency is satisfied if the local constraints are satisfied.

We partition the domain of discourse into *plan instances* and *attribute values,* which are denoted by PI and AV respectively. *Attributes* are functions which maps plan instances PI to attribute values. For example, *time* is an attribute which maps a plan instance to the time interval representing its duration. The set of attributes is denoted by *A.*

A plan type is a unary predicate on PI (or equivalentjy, a subset of PI). We use the symbol *P* to denote the set of plan types. For example, MkNdl (Make Noodle) and Boil are two plan types. The relationships between a plan and its immediate components are represented by *features,* which are functions from PI to PI. For example, that Boil is an immediate component of MkNdl is represented by a feature f_1 which maps an instance of MkNdl into an instance of Boil. The set of features is denoted by *F.*

The symbol $\perp$ denotes the void element in **PI**. For an element $x \in$ **PI**, $f(x) = \perp$ means that x does not have an f-component. We defined $p(\perp)$ to be false and $f(\perp) = \perp$, for any $p \in P$, and $f \in F$.

Plans are axiomized by three sets of FOL formulas: Abstraction Axioms, Feature Restriction Axioms, Attribute Value Constraints.

Definition 2.1 (Abstraction Axioms) *An abstraction axiom is a formula of the form:* $\forall x.p_1(x) \rightarrow p_2(x)$, *where* $p_1, p_2 \in P$.

The following are two examples of abstraction axioms:

$\forall x.$**MkSpagPesto**$(x) \rightarrow$ **MkPasta**(x)

$\forall x.$**MkSpag**$(x) \rightarrow$ **MkNdl**(x)

Corresponding to each abstraction axiom $\forall x.p(x) \rightarrow q(x)$, there is a specialization function q_spec_p such that

$$q\_spec\_p(x) = \begin{cases} x & \text{if } p(x) \text{ is true} \\ \perp & \text{otherwise} \end{cases}$$

The specialization functions make the functional inverse of abstraction axioms explicit.

Definition 2.2 (Feature Restriction Axioms) *A feature restriction axiom is a formula of the form:*

$$\forall x.p(x) \wedge (f(x) \neq \perp) \rightarrow p'(f(x))$$

where $p, p' \in P$ *and* $f \in F$.

We write $p \xrightarrow{f} p'$ as a short hand for this axiom. Intuitively, each plan type p has an associated set of feature restriction axioms which define the possible components of each instance of p. The condition $f(x) \neq \perp$ ensures that the plan type of a component is specified only when a component exists. For example, suppose f_1 is a feature (function) that maps an instance of **MkNdl** to a component, say its second step, an instance of **Boil**. Then there is a feature restriction axiom $\forall x.$**MkNdl**$(x) \wedge (f_1(x) \neq \perp) \rightarrow$ **Boil**$(f_1(x))$.

A feature $f \in F$ is said to be a *feature of a plan type* $p \in P$ provided that there exists a feature restriction axiom $\forall x.p(x) \wedge (f(x) \neq \perp) \rightarrow p'(f(x))$. For example, f_1 is a feature of **MkNdl**, because

$$\forall x.\textbf{MkNdl}(x) \wedge (f_1(x) \neq \perp) \rightarrow \textbf{Boil}(f_1(x))$$

is a feature restriction axiom.

If a plan instance x belongs to plan type p and f is a feature of p (or a superclass of p), then $f(x)$ is an *immediate component* of x. The *components of a plan instance* x are the plan instances that are either immediate components of x or, recursively, components of the immediate components of x.

Definition 2.3 (Attribute Value Constraints) *There are two types of attribute value constraints: local constraints and percolation constraints.*

A Local Constraint *specifies how the attributes of a plan instance are related, and is expressed by a formula of the form:*

$$\forall x.p(x) \rightarrow r(a_{i_1}(x), a_{i_2}(x) \ldots, a_{i_n}(x))$$

where $p \in P$, r *is a n-ary predicate over attribute values* (**AV**) $(n = 1, 2, \ldots)$.

A Percolation Constraint *specifies how the attributes of an plan instance are determined by the attributes of its immediate components and is a formula of the form:*

$$\forall x, v \;.\; p_1(x) \wedge p_2(f(x)) \wedge a'(f(x)) = v \;\rightarrow\; a(x) = v$$

where $p_1, p_2 \in P$, f *is feature of* p_1 *and* $a, a' \in A$ *are attributes.*

Suppose *time*, $time\_f_1$, and $time\_f_4$ are attributes and *before* is a binary predicate on **AV**, then

$\forall x.$**MkSpag**$(x) \rightarrow before(time\_f_4(x), time\_f_1(x))$

is an example of local constraints.

Suppose f_4 is a feature of **MkSpag** that maps an instance of **MkSpag** to its **GetSpag** component, and f_1 is a feature of **MkNdl**, two examples of percolation constraints are:

$$\begin{aligned} \forall x, v. \quad & \textbf{MkSpag}(x) \wedge \textbf{GetSpag}(f_4(x)) \wedge time(f_4(x)) = v \\ & \longrightarrow time\_f_4(x) = v \\ \forall x, v. \quad & \textbf{MkNdl}(x) \wedge \textbf{Boil}(f_1(x)) \wedge time(f_1(x)) = v \\ & \longrightarrow time\_f_1(x) = v \end{aligned}$$

In summary, the knowledge about plans is represented by three sets of FOL formulas. We write **KB** to denote the conjunction of the set of abstraction axioms, feature restriction axioms and the attribute value constraints.

3 Scenarios

A plan recognizer's task is to find an explanation of the observed actions. We define explanations to be consistent scenarios that entail the observations. A scenario is an FOL formula that describes an plan instance and its components. The syntax of a scenario is not defined by BNF, rather, it is defined in terms of walk trees in the plan hierarchy which is the diagrammatic form of the abstraction and feature restriction axioms in KB.

Definition 3.1 (Plan Hierarchy) *A plan hierarchy t$ a directed graph whose nodes represent elements tn P and whose links are as follows:*

"isa" and specialization links: *Corresponding to each abstraction axiom* $\forall x.p_1(x) \rightarrow p_2(x)$ *in* I, *there is an "isa" link* $p_1 \xrightarrow{isa} p_2$, *and a specialization link* $p_2 \xrightarrow{spec} p_1$.

feature links: *Corresponding to each feature restriction axiom:*

$$\forall x.p_1(x) \wedge (f(x) \neq \perp) \rightarrow p_2(f(x))$$

there is a feature link *from* p_1 *to* p_2 *and labeled f.*

We write *H* to denote the plan hierarchy for KB. The sets of "isa" links and feature links correspond to the abstraction and decomposition hierarchies in [Kautz, 1987] respectively. Figure 1 shows the plan hierarchy of a cooking world. Each of the bidirectional gray arrows represents a pair of "isa" and specialization links. The upward arrow represents the "isa" link and the downward arrow represents the specialization link.

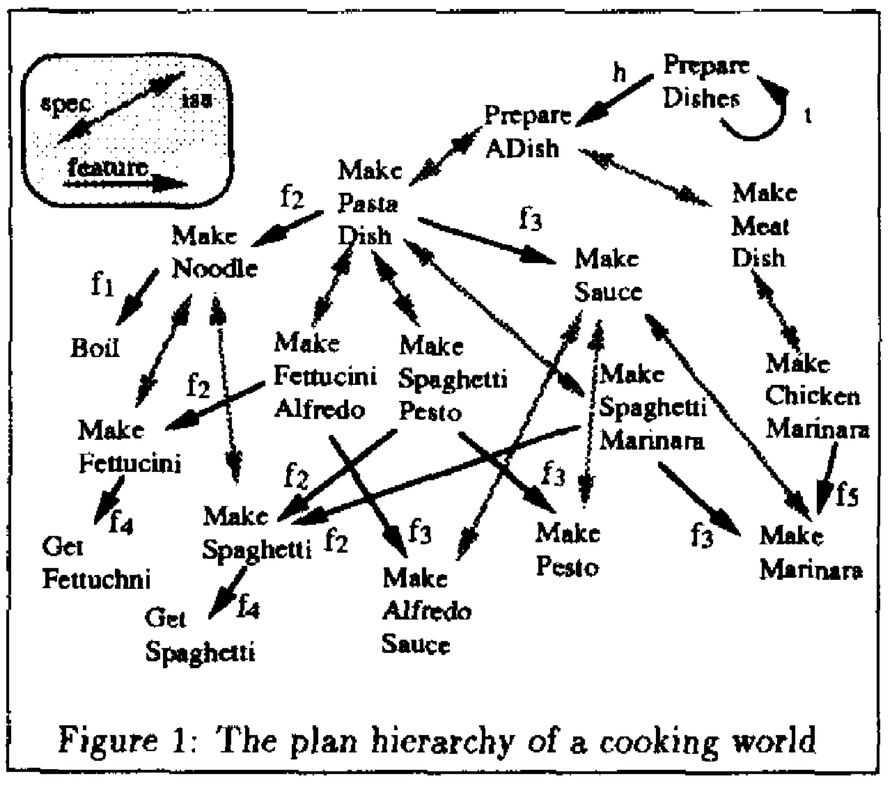


Figure 1: The plan hierarchy of a cooking world

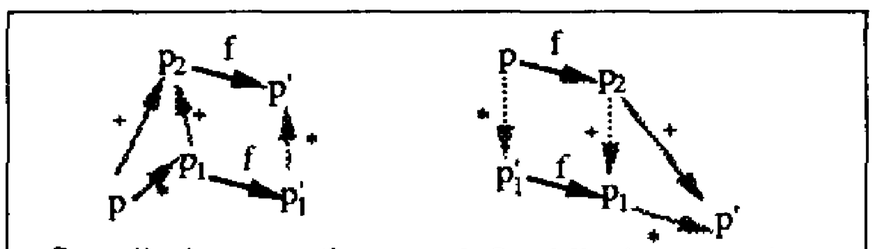
The "isa" links (as well as the specialization links) induce a partial order ("isa" relation) over P. The closure of "isa" relation is denoted by $\xrightarrow{isa}*$ and the non-empty closure is denoted by $\xrightarrow{isa}+$. The meaning of $\xrightarrow{spec}*$ and $\xrightarrow{spec}+$ are similarly defined.

A directed path in the plan hierarchy $\mathcal{H}$ represents a plan–component relationship between the two end nodes of the path. The "isa" links allow plan types to *inherit arguments* from their superclasses. For example, The path from **MkPasta** to **Boil** is an argument that an instance of **MkPasta** has an instance of **Boil** as one of its components. This argument can be inherited by MkSpagPesto via the "isa" link from it to MkPasta The specialization links, on the other hand, allow plan types to *inherit explanations* from their superclasses. For example, an explanation of a MkPasta instance, together with an assumption that the MkPasta instance is a MkSpagPesto instance, may also serve as an explanation of the MkSpagPesto instance, provided that there does not exist a more direct explanation for the latter. Similar to inheritance reasoning [Touretzky, 1986; Bacchus, 1988], path preemption ensures that only the most specific information is inherited -

Definition 3,2 (Path Preemption)

Generalization Preemption: *A path* $p \xrightarrow{isa}+p_2 \xrightarrow{f} p'$ *is preempted if there exists a path* $p \xrightarrow{isa}*p_1 \xrightarrow{f} p_1' \xrightarrow{isa}*p'$ *such that* $p_1 \xrightarrow{isa}+p_2$ *(Figure 2.a).*

Specialization Preemption: *A path* $p \xrightarrow{f} p_2 \xrightarrow{spec}+p'$ *is preempted if there exists a path* $p \xrightarrow{spec}*p_1' \xrightarrow{f} p_1 \xrightarrow{spec}*p'$ *such that* $p_2 \xrightarrow{spec}+p_1$ *(Figure 2.b).*



a. Generalization preemption b. Specialization preemption

Figure 2: Path Preemption

An observed action can be related to some higher *level* plan by a path, which identifies the action as a possible component of achieving the plan. The plan recognizer's task, however, is not just to explain the observed actions separately. Rather, it must find a set of coherent relationships which relate all the observations to a higher level plan. To accomplish this, we introduce the notion of walk tree. The definition of a walk tree is built recursively upon the concept of a local tree, which identifies the relationships between a plan and its immediate components.

Definition 3.3 (Local Tree) *A local tree of H is a directed subtree T of H such that the paths from the root of T to the leaves of T are sequences of zero or more "isa" links followed by one feature link and the labels of the feature links are different.*

An example *of* Local Tree is shown in Figure 3.a

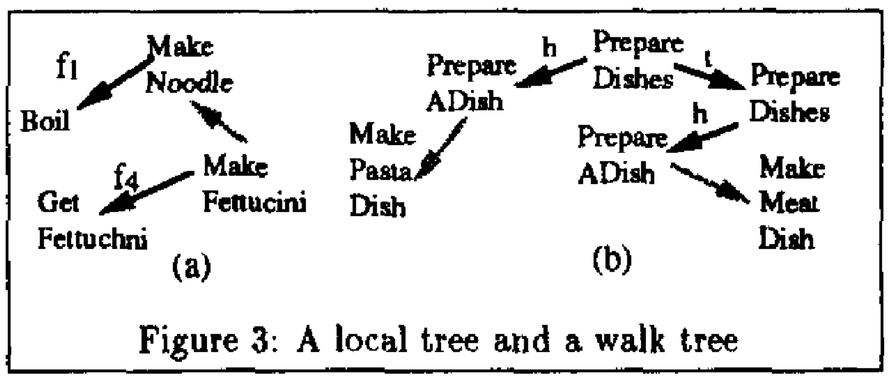


Figure 3: A local tree and a walk tree

Definition 3.4 (Walk Tree) *A walk tree of the plan hierarchy* $\mathcal{H}$ *is defined recursively as follows:*

1. *The local trees of $\mathcal{H}$ are also walk trees of $\mathcal{H}$.*
2. *Suppose α is a walk tree of $\mathcal{H}$. If a leaf node of α is labeled p and β is a local tree or a specialization link in $\mathcal{H}$, with the root labeled p, then the tree formed by joining the node p in α and the root of β is also a walk tree of $\mathcal{H}$.*
3. *All walk trees are either one of the above.*

A walk tree is said to be *valid* if it does not contain any preempted paths. A walk tree is more general than a tree in that there may be multiple occurrences of the same node in $\mathcal{H}$, and therefore can be used to represent iterative or recursive plan components. For example, Figure 3.b shows a valid walk tree, where there are two instances of **PrepareADish**.

An *attribute assignment* of a plan instance x is a set of assignments of attribute values to x and is denoted by $V(x)$. That is $V(x) \equiv \bigwedge_{i=1}^{k} a_i(x) = v_i$, where $a_i \in A$ and v_i's are constants in **AV**. For example, $time(x) = [0,1] \wedge agent(x) = john$ is an attribute assignment of x.

Definition 3.5 (Scenario) *Let T be a valid walk tree of $\mathcal{H}$. Let $f_1^*, f_2^*, \ldots f_m^*$ be the compositions of the feature and specialization functions along the path from the root of T to the leaf nodes of the local trees in T. Let $p_1, p_2, \ldots p_m$ be the corresponding plan types represented by the nodes. Then T and a set of attribute assignments $V_1, V_2, \ldots, V_m$ define a scenario α, which is an FOL formula:*

$$\alpha \equiv \exists x. \bigwedge_{i=1}^{m} (p_i(f_i^*(x)) \wedge V_i(f_i^*(x))).$$

For example, suppose *time* is an attribute of plans. Let α denote the scenario in Figure 4, where the intervals beside the nodes denote the values of their time attribute. Then α is the following formula:

$$\begin{aligned}\alpha \equiv \exists x\, & \mathbf{MkSpagMari}(x) \wedge time(x) = [0,2] \wedge \\ & \mathbf{MkMari}(x) \wedge time(f_3(x)) = [1,2] \wedge \\ & \mathbf{MkSpag}(f_2(x)) \wedge time(f_2(x)) = [0,2] \wedge \\ & \mathbf{GetSpag}(f_4(f_2(x))) \wedge time(f_4(f_2(x))) = [0,1] \wedge \\ & \mathbf{Boil}(f_1(f_2(x))) \wedge time(f_1(f_2(x))) = [1,2]\end{aligned}$$

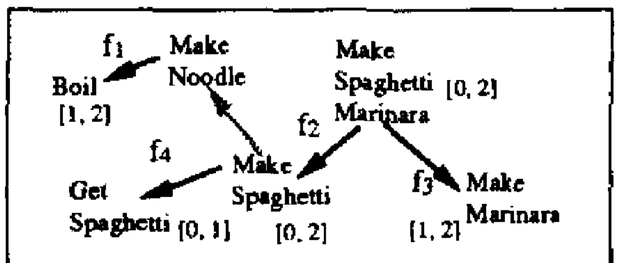


Figure 4: A scenario of **MkSpagMari**

4 The Plan Recognition Problem

An *observation* is an assertion that there exists a plan instance of certain type and possessing certain attributes.

Definition 4.1 (Observation) *An observation is a formula of the form:*

$$o \equiv \exists x. p(x) \wedge V(x).$$

where p is a plan type and V is an attribute assignment.

Definition 4.2 (Explanation) *Given* **KB** *and a set of observations $o_1, \ldots, o_k$, an explanation of the observations is a scenario α such that*

1. *α is consistent with* **KB**, *i.e.*, $\mathbf{KB} \wedge \alpha \not\models false$.
2. $\mathbf{KB} \wedge \alpha \models o_1 \wedge \ldots \wedge o_k$.

Definition 4.3 (Plan Recognition Problem) *Given* **KB** *and a set of observations $\{o_1, \ldots, o_k\}$, the plan recognition problem is to find the most general explanation α of $\{o_1, \ldots, o_k\}$. That is, there does not exist another explanation β of $\{o_1, \ldots, o_k\}$ such that $root_\alpha \xrightarrow{isa}{+} root_\beta$, where $root_\alpha$ and $root_\beta$ are the plan type represented by the root nodes of α and β respectively.*

The more general explanation is preferred because a more general statement about a plan instance is more likely to be true. For example, $\mathbf{MkPasta}(x)$ is more likely to be true than $\mathbf{MkSpagPesto}(x)$. On the other hand, the validity of the walk tree implies that each explanation contains only the arguments that are the most specific with respect to the observations.

4.1 Examples

Suppose the Abstraction Axioms and Feature Restriction Axioms are represented by Figure 1. The Percolation Constraints are as follows:

$$\begin{aligned}&\forall x, v. \mathbf{MkNdl}(x) \wedge \mathbf{Boil}(f_1(x)) \wedge time(f_1(x)) = v \\ &\qquad \longrightarrow time\_f_1(x) = v \\ &\forall x, v. \mathbf{MkSpag}(x) \wedge \mathbf{GetSpag}(f_4(x)) \wedge time(f_4(x)) = v \\ &\qquad \longrightarrow time\_f_4(x) = v \\ &\forall x, v. \mathbf{MkPasta}(x) \wedge \mathbf{MkNdl}(f_2(x)) \wedge time(f_2(x)) = v \\ &\qquad \longrightarrow time\_f_2(x) = v \\ &\qquad \ldots\ldots\end{aligned}$$

The Local Constraints are as follows:

$$\forall x. \mathbf{MkNdl}(x) \rightarrow before(time\_f_4(x), time\_f_1(x))$$
$$\forall x. \mathbf{MkNdl}(x) \rightarrow time(x) = union(time\_f_4(x), time\_f_1(x))$$

Example 1: Suppose the following observations are made:

$o_1 \equiv \exists x. \mathbf{GetSpag}(x) \wedge time(x) = [0,1]$
$o_2 \equiv \exists x. \mathbf{Boil}(x) \wedge time(x) = [1,2]$
$o_3 \equiv \exists x. \mathbf{MkMari}(x) \wedge time(x) = [1,2]$

Then the scenario that explains $\{o_1, o_2, o_3\}$ is the one shown in Figure 4. Note that the observations do not have to be a sequence of actions. In this example, **Boil** and **MkMari** are concurrent.

Example 2: Suppose the following two observations are made:

$o_1 \equiv \exists x. \mathbf{MkNdl}(x) \wedge time(x) = [0,1]$
$o_2 \equiv \exists x. \mathbf{MkSauce}(x) \wedge time(x) = [1,2]$

The explanations of $\{o_1, o_2\}$ are shown in Figure 5. Since the explanation in 5.a is more general than those in 5.b, 5.c and 5.d, the solution of the plan recognition problem is the scenario in 5.a.

Example 3: Suppose the observations are:

$o_1 \equiv \exists x. \mathbf{MkFett}(x) \wedge time(x) = [0,1]$
$o_2 \equiv \exists x. \mathbf{MkAlSauce}(x) \wedge time(x) = [1,2]$

Then the only explanation is the scenario in Figure 5.c. The walk tree shown in Figure 6 is not a valid one because the path:

$$\mathbf{MkPasta} \xrightarrow{f_2} \mathbf{MkNdl} \xrightarrow{spec} \mathbf{MkFett}$$

is preempted by the path

$$\mathbf{MkPasta} \xrightarrow{spec} \mathbf{MkFettAl} \xrightarrow{f_2} \mathbf{MkFett}.$$

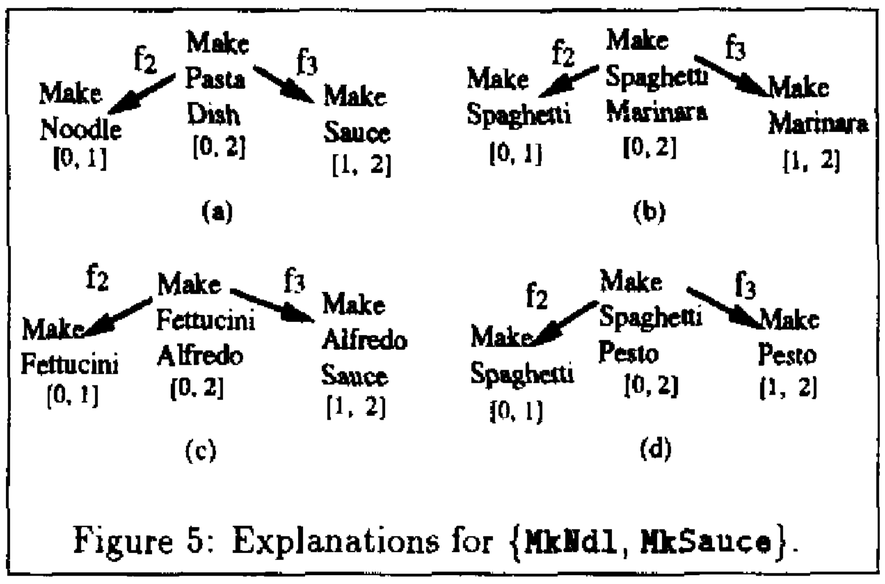


Figure 5: Explanations for {MkNdl, MkSauce}.

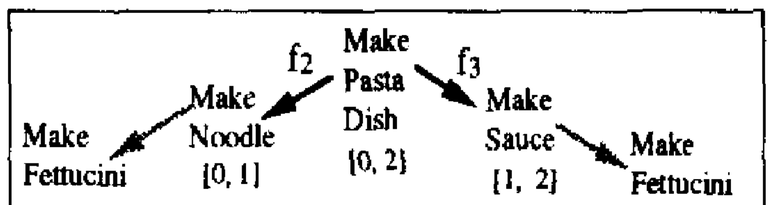


Figure 6: An invalid walk tree

5 A Recognition Algorithm

Since only a restricted subset of FOL is used to represent the observations, scenarios, as well as the domain knowledge, the plan recognition problem may be solved by a specialized inference method. We now present a message passing algorithm for plan recognition. Each node in the plan hierarchy is a computing agent which communicates with other agents by sending messages across the links in the plan hierarchy. A message is a pair: <V, B>, where V is an attribute assignment describing a plan instance, and B is a subset of observations. The messages are passed in the reverse direction of the links in the plan hierarchy. If a message <*V,B*> reaches a node *p,* then this means that the observations in B can be explained as components of an instance of plan type *p* that matches the description V, Therefore, the messages represent partial explanations of subsets of the observations. The partial explanations are combined at the nodes to generate explanations for larger subsets of the observations.

Let $\{o_i \equiv \exists x.p_i(x) \wedge V_i(x)\}$ $(i = 1, 2, \ldots, n)$ be a set of observations. The message passing process is initiated by sending the message $<V_i, \{o_i\}>$ to the node p_i, for $i = 1, 2, \ldots, n$. The message passing process stops when no further messages are being sent and the explaining scenarios can then be retrieved from the network by tracing the origins of the messages whose B components are equal to { O_1, O_2,.....,O_n }.

When a node receives a message via a specialization link, the message is forwarded via the incoming specialization and feature links. When a node receives a message <*V,B*> via an "isa" link or a feature link, the message is first forwarded across the incoming "isa" links, then an item is created and saved into the local memory. An item is a triple <V, B, f>, where V and B are the same as in messages, and F is a set of features. The item created when the message <V, *B*> arrives is <*V', B, {f}*>, where *V'* is determined by V according to the percolation constraints and / is the feature represented by the last feature link traversed by the message. The node then attempts to combine the new item with previously stored items that are compatible with it. Two items <V_1, B_1, F_1> and <V_2 B_2, F_2> are compatible if F_1 and F_2 are disjoint and the unification of $V\backslash$ and V_2 satisfies the local constraints. If interleaved plan components are not allowed, B_1 and B_2 must also be consecutive sequences of observations. If shared plan components are not allowed, then the two items are only compatible if $B\backslash \cap B_2 = \emptyset$.

The combination of two items <V_1,B_1,F_1> and <V_2, B_2, F_2> is an item <V_0,B_o,F_o>, where Vb is the unification of V_1 and V_2; B_0 - $B_1 \cup B_2$ and F_0 = $F_1 \cup F_2$. If <*Vo,Bo,Fo*> satisfies all the local constraints, the message <V_o,Bo> is sent further if no identical messages have been previously sent by the current agent.

The message passing algorithm must ensure that the scenarios be consistent with the knowledge base and their walk tree is a valid one, i.e., they must not contain preempted walks. The consistency of the explanations is maintained by checking the percolation and local constraints during the message passing process. The validity of the walk trees of the scenarios is guaranteed by blocking certain messages.

We define a binary relation *preempted* as follows: Let a be a feature link $p \xrightarrow{f} q$ and *b* be a node in H. *precmpted(a, b)* is true iff 1) the path $b \xrightarrow{isa} p \xrightarrow{f} q$ is preempted; or 2) the path $p \xrightarrow{f} q \xrightarrow{spec} b$ is preempted.

The relation *preempted* is precomputed and remains unchanged as long as the plan hierarchy is unchanged. A message sent across a "isa" link $c \xrightarrow{isa} b$ is blocked if *preempted(a,b)* is true, where *a* is the last feature link traversed by the message. A message sent across a feature link *a* is blocked if *preempted(a,b)* is true, where 6 is the last node traversed by the message before reaching *a* via a sequence of specialization link.

The message passing algorithm is shown in Figure 7. The utility functions used in the algorithm are listed and explained in Table 1. A more detailed description of the algorithm as well as its correctness proof can be found in [Lin and Goebel, 199l].

Since no identical messages are sent across a link more than once, the total number of messages is bounded by 0(M|7i|), where \H\ is define to be the number of links in H and *M* is the number of distinct messages. The total number of items created is bounded by $0(M\backslash H\backslash 2^d)$, where *d* is the maximum number of immediate components a plan has.

Suppose the same set of restrictions is imposed on the plan hierarchy as in [Vilain, 1990], that is, plan components are linearly ordered and cannot be shared or interleaved. Then the plan recognition algorithm degenerates to a message passing algorithm for context-free grammars [Lin and Goebel, 1990), which has complexity $0(n^3|H|)$, where n is the number of observations. This is a significant improvement over [Vilain, 1990] which has the complexity $0(n^3\backslash H\backslash^2)$ because $|\mathcal{H}|$ tends to be a large number in non-trivial domains.

Algorithm 1 Send Message

```
sendViaFeatureLink(Node sender, Message m, Node n)
  foreach feature link l of sender do
      if (not preempted(l, n)) receive(tailOf(l), m, l);
end

sendViaIsaLink(Node sender, Message m, FeatureLink f)
  foreach isa link l of sender do
      if (not preempted(f, tailOf(l))) receive(tailOf(l), m, l);
end

sendViaSpecLink(Node sender, Message m, Node n)
  foreach specialization link l of sender do
      receiveViaSpec(tailOf(l), m, n);
end
```

Algorithm 2 Receive Message

```
receive(Node receiver, Message m, Link b)
  if (isNew(receiver, m)) sendViaIsaLinks(receiver, m, b);
  i ← item(m, b);
  forall items i' such that compatible(receiver, i', i) do
    i'' ← combine(receiver, i, i');
    if (not complete(receiver, i'')) continue;
    m' ← message(i'').
    if (not isNew(receiver, m')) continue;
    sendViaFeatureLinks(receiver, m', nil);
    sendViaSpecLinks(receiver, m', receiver);
  end
end

receiveViaSpec(Node receiver, Message m, Node b)
  if (isNew(receiver, m))
    sendViaSpecLinks(receiver, m, b);
    sendViaFeatureLinks(receiver, m, b);
  end
end
```

Figure 7: A Message Passing Algorithm

Table 1: Utility functions

Function*	Return value
combine(n, i, i');	the combination of i and i'
compatible(n, i, i');	true if i and i' are compatible at node n.
complete(n, i);	true if i satisfies all the local constraints;
isNew(n, m);	true if m has not previously been sent by n;
item(m, l);	the item $<V, B, f_l>$, where $<V, B>$ = m and f_l is the feature represented by l.
message(i);	the message $<V, B>$, where $<V, B, F>$ = i.

*n is a Node; i and i' are Items; m is a Message; l is a Link;

If the plan components are partially ordered, but are not shared or interleaved, the complexity of our plan recognition algorithm is the same as the complexity of direct ID (Immediate Dominance) rule parsing in [Lin and Goebel, 1990], i.e., $0(n^2|H|2^d)$, where d is the maximum number of features of a plan type.

6 Conclusion

We have presented a message passing algorithm for plan recognition, which has an asymptotic upper bound that is an improvement on previous related work. Our representation scheme is more general than Kautz's in its ability to accommodate optional and recursive plan components, but more restrictive in the pattern of constraints that can be specified. The logical representation of plans is made computationally explicit, as a graph structure in which plans are manifest as a special kind of graph walk. Explanations of the observations are defined to be valid scenarios that are consistent, with the knowledge base and, together with knowledge base, logically entail the observations. A scenario is valid if it consists only of arguments that are the most specific with respect to the observations. In case of multiple possible explanations, the most general one is preferred over the others.

Acknowledgement

Peter van Beek has read two drafts of this paper and provided valuable comments. Thanks are also due to critics and suggestions from anonymous reviewers. This research is done while the first author is at the Department of Computing Science, University of Alberta,

References

[Bacchus, *1988]* Fahiem Bacchus. *Representing and Reasoning with Probabilistic Knowledge.* PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 1988.

[Goodman and Litman, 1990] B. Goodman and D. Litman. Plan recognition for intelligent interfaces. In *Proceedings of the IEEE Conference on Artificial Intelligence Applications,* pages 297-303, 1990.

[Kautz, 1987] Henry A, Kautz. A formal theory of plan recognition. Technical Report 215, Department of Computer Science, University of Rochester, May 1987.

[Lin and Goebel, 1990] Dekaug Lin and Randy Goebel. Context free grammar parsing by message passing, (Unpublished), 1990.

[Lin and Goebel, 199l] Dekang Lin and Randy Goebel. A message passing algorithm for plan recognition. Technical Report 91-7, Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada, 1991.

[Litman and Allen, 1991] D. Litman and J. Allen. Discourse processing and commonsense plans. In *Intensions in Communication.* MIT Press, 1991.

[Lochbaum *et al.,* 1990] Karen E, Lochbaum, Barbara J. Grosz, and Candace L. Sidner. Models of plans to support communication: an initial report. In *Proceedings of AAAI-90,* pages 485-490, 1990.

[Touretzky, 1986] David Touretzky. *The Mathematics of Inheritance Systems.* Morgan Kaufmann, 1986.

[Vilain, 1990] Marc Vilain. Getting serious about parsing plans: a grammatical analysis of plan recognition. In *Proceedings of AAAI-90,* pages 190-197, Boston, MA, July 1990. The MIT Press.