A Knowledge–Based Approach to Goal Recognition

Ahmed Guessoum Torulf Mollestad

Knowledge Systems Group
Department of Computer Systems and Telematics
The Norwegian Institute of Technology
7034 Trondheim, Norway
emails: {ahmed,torulf}@idt.unit.no

Abstract

We tackle in this paper the problem known as plan recognition. Our aim is to try to restudy the problem, paying particular attention to the underlying reasoning processes. We rename the problem as *goal recognition*, explain the details of the reasoning steps that are required for it, and argue in particular that forward reasoning is the central process in goal recognition though it needs to be supported by other forms of reasoning such as abduction. We also sketch algorithms for the implementation of the various processes, give an integrated algorithm, and illustrate it with various examples that highlight some of the problems that can be encountered such as that of inconsistency between the observations and the available knowledge.

1 Introduction

During the last decade, a great deal of effort was devoted to Natural Language Understanding (computer) systems. In this endeavour, researchers have tried to analyse and model the way human beings communicate. Of particular interest was the problem of finding out what a dialogue partner means, including what he/she aims at by making a certain dialogue move. One aspect of this problem, which consists in finding out the underlying plan/goal that motivated an agent's actions, has classically been called *Plan Recognition*. Earlier work on Plan Recognition was based on an attempt to encode as many plans as possible and to match observed "actions" against these plans. Allen [All83] based his work on such plans, called scripts. Wilensky [Wil83] also followed this approach but argued that plan recognition systems should reason not only about how actions achieve domain goals, but also about how the selection and ordering of actions is influenced by higher level planning goals such as minimising resource expenditure. Litman [Lit85] later developed these ideas further. Sidner [Sid83] was also inspired by Allen's work. Though her approach was similar, she enriched her model with information about what a hearer knows about the speaker's actions as well as the hearer's knowledge about his/her own beliefs. Litman and Allen [LA87] describe a plan-based model for dealing with subdialogues in conversations..

More recently, some effort has been directed to a more logic-based formulation of the problem and its solutions. Helft and Konolige [HK90] suggested a solution to the plan recognition problem which is based on abduction ([Fan70, Eis79]) and the notion of consistency. Appelt and Pollack [AP92] suggested the use of *weighted abduction* ([HS90]). Kautz' account of plan recognition [Kau87, Kau90] relies on the use of a form of circumscription ([McC80, McC86]) to express various axioms about his knowledge bases. His algorithms would return, for a given set of observations, all the goals that may explain them. Probabilistic solutions to the plan recognition problem have also been suggested, e.g. by Callistri-Yeh [CY91], Charniak and Goldman [CG93] and Raskutti and Zukerman [RZ91].

Despite the fact that plan recognition has been studied quite thoroughly, we believe that the subject still requires further work. One problem is that the terminology used in the area is not uniform. For instance, whereas "plan recognition" in [Kau87, Kau90], [AP92], and others means the recognition of *goals* that explain a given set of observations, the same term is used by Sidner to denote the generation of plans that fulfill a stated goal. (The latter problem is usually known as *planning* [MH69, FN71, Sac75].) Moreover, Konolige [Kon94] suggests two different terms: "simple plan recognition" which is similar to Sidner's "plan recognition" (i.e. planning) and "complex plan recognition" which is similar to the former meaning of "plan recognition". Obviously, this disagreement over the terminology is bound to have an effect on the solutions that are suggested to the plan recognition problem. Another, even more serious problem, is that plan recognition—as a reasoning process–still needs some attention. Different authors have recently argued on the nature of "plan recognition". Some supported that it is abductive (e.g. [AP92]) while others have argued that it is rather a deductive process (e.g. [Kau87, Kau90]). A disagreement over the nature of "plan recognition" is also bound to affect the solutions that are put forward.

In this paper, we try to restudy the problem of "plan recognition", taking into account the issues mentioned above. First, we suggest the use of the term *goal recognition* rather than "plan recognition" since the former term better emphasises the fact that an observer is mostly interested in the goals of the agent; the plans carried out to achieve these goals are generally of secondary importance. We will then argue that "plan recognition" is *not* abductive, but intrinsically based on forward reasoning (*not* deduction) processes. Abduction will, however, play a "supporting role" in our solution to the problem.

Let us now give some informal definitions of the terminology used in the rest of this paper. *A Goal* describes some intended/desired state; it is the underlying motivation behind an agent's actions and utterances. *A Plan* (relative to a goal and a task domain) is a sequence/set of actions which, when performed in the task domain, will achieve the specific goal. *Planning* is the process of finding a plan to realise a given goal. *An Observation* is a piece of information that either (1) some event has taken place or (2) that the event has not *and will not* take place in the current session. We call the observation *positive* in the first case, and *negative* in the latter. Given a set of observations (events or actions carried out by some agent), *Goal Recognition* is the problem of drawing conclusions as to the underlying goal or goals that motivated the agent's actions. In this paper we study the problem of keyhole recognition, where we do not make the assumption that the agent is structuring his/her actions in a way that is meant to ease the process of plan recognition.

Let us consider a short example to illustrate the problem. Assume that a plan for taking the train to a destination D comprises three subactions each of which has to be performed in order for the overall goal to be realised. Assume that these actions are, *goto(P,station)*, *buy_ticket(P,D)*, and *board_train(P,D)*. If we now observe an agent performing *buy_ticket(agent1,oslo)*, we want to be able to ascribe to this agent the goal *take_train(agent1,oslo)*. Such reasoning from an observation to a plausible underlying goal is what we call *goal recognition*.

Goal recognition is used in different settings. For instance, dialogue systems should be able to recognise the intentions of a user (see for instance [All83, Kau87, Kau90, Mol89] and the ESPRIT Project PLUS (P5254)). This feature is important in cooperative systems as well so that they be able to correct user misconceptions, suggest alternative plans and provide not explicitly requested, though helpful, information. The principle of plan recognition has also been used within different areas, most notably in work on story understanding, natural language pragmatics and intelligent database interfaces.

In the next section, we discuss some general issues in knowledge representation and present our knowledge representation framework and an example of knowledge bases which will be used later in the paper. In Section 3, we explain the basic reasoning processes that underlie goal recognition. This is followed in Section 4 by algorithms to implement our solution to the problem of goal recognition and the issue of incrementality of observation processing is mentioned. A discussion of some related work is given in Section 5, followed by a comparison to our work. We conclude in Section 6.

2 Knowledge Representation

2.1 General Issues in Knowledge Representation

It is our belief that a clear separation between knowledge and reasoning is vital to getting a good understanding of the problem of goal/plan recognition. The knowledge should be structured in an intuitive manner, without introducing extralogical features such as, for instance, the end–event and its intended use in Kautz' work. The algorithm is then defined in terms of basic modes of reasoning, independently from the particular knowledge representation scheme chosen for the application at hand.

Furthermore,the knowledge should be separated into two distinct subparts, each representing different relations between the defined entities (*concepts* in our terminology):

- **the Ontology**, representing the hierarchy of concepts, i.e. the relationship between all the concepts and their subconcepts;
- **the Plans Knowledge Base**, representing the relationship between a goal and the concepts that may be part of a plan to realise this goal. Knowledge of the form "performing s_1 and s_2 ... and s_n will achieve g" is encoded in the Plans Knowledge Base.

2.2 Logic-Based Knowledge Representation

The knowledge representation principles just explained form the basis of our knowledge representation. Ultimately, it is a clear separation of "knowledge" and "reasoning" that is sought. It should therefore be clear that the solution to the goal recognition problem which we present below is not dependent on the particular representation formalism adopted. Nevertheless, we have seen fit—for the purposes of explaining our solutions and selecting a suitable knowledge representation framework—to adopt a logic programming approach.

In performing goal recognition, a person or system needs to have knowledge of plans that achieve various goals. For instance, one may know that one way to "watch a movie" is to "go to a cinema", "buy a ticket", and "take a seat in the projection room". In other words, if one performs these three steps, one will (be able to) watch a movie. This can be written as

$$\textit{watch(P,movie)} \leftarrow_{\mathrm{d}} \textit{goto(P,cinema)} \wedge \textit{buy(P,ticket)} \wedge \textit{sit\_in(P,projection\_room)}$$

This is obviously a schematic writing of a plan. For instance, one would need to represent "cinema", "ticket", and "projection_room" as objects whose type will be the concepts "cinema", "ticket", and "projection_room", respectively. So

$$goto(P, cinema)$$

could be written as:

$$goto(P, Place) \wedge cinema(Place)$$

It is important to note that the arrow "$\leftarrow_{\mathrm{d}}$" is used to denote a *default* rule, not an implication. Indeed, it may be that a cinema inspector, for instance, goes to a cinema, (incidentally) buys a ticket for a friend, and sits in the projection room (as part of his/her inspection). His/her goal is clearly not necessarily to watch a movie.

More generally, it is natural to represent a plan whose steps are "s_1", ..., "s_n" to achieve a goal "g", as a rule of the form $g \leftarrow_{\mathrm{d}} s_1 \wedge \ldots \wedge s_n$.

On the other hand, plans are defined in terms of concepts in the world. These concepts can be related as subconcepts (or superconcepts) of one another. They are all part of a world ontology which can be represented in a structure[1], where two concepts are linked if one is a subconcept of the other. For

[1] Although we consider single–inheritance only, our approach would cope just as well with the multiple–inheritance

instance, one can think of the part of the ontology which describes the action of "buying" as shown in Figure 1.

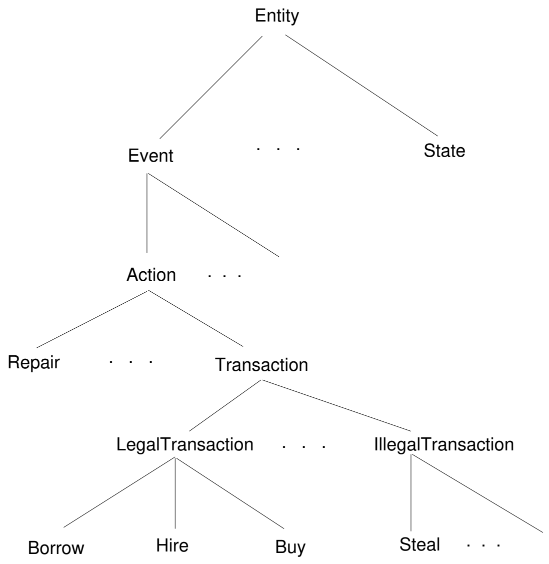

Figure 1: An example of a concept hierarchy

Obviously, in such a hierarchy of concepts, whenever a concept holds, then its superconcept also holds. For instance, if any act is a "Buy", then it is also—by definition—a "LegalTransaction". Hence, one can represent such a concept hierarchy as a set of definite clauses whose body has exactly one atom. For the above hierarchy, we have the following clauses:

$LegalTransaction(X) \leftarrow Buy(X)$
$LegalTransaction(X) \leftarrow Hire(X)$
$LegalTransaction(X) \leftarrow Borrow(X)$
$IllegalTransaction(X) \leftarrow Steal(X)$
$Action(X) \leftarrow Transaction(X)$

and so on.

More generally, if A and B are two concepts in a concept hierarchy such that A is a superconcept of B (or, equivalently, B is a subconcept of A), then this relation is represented as a clause $A(t_1) \leftarrow B(t_2)$, where t_1 and t_2 are the vectors of arguments (or properties) assigned in the hierarchy to A and B, respectively.

The above shows that the logic programming framework is suitable for representing plans and concept-hierarchies.

2.3 A Selected Example

In the preceding sections we have introduced our knowledge representation scheme, and given the motivation behind our choices. In the sequel, we will make use of the "Cooking World" knowledge base [Kau87, Kau90]. Even though this example is propositional, the subsequent explanations will show that our algorithm for goal recognition is general enough to handle the non–propositional case.

---

case.

The (Cooking World) Hierarchy:

$$\begin{array}{lll} PrepareMeal & \leftarrow & MakeMeatDish \\ PrepareMeal & \leftarrow & MakePastaDish \\ MakePastaDish & \leftarrow & MakeFettuciniAlfredo \\ MakePastaDish & \leftarrow & MakeSpaghettiPesto \\ MakePastaDish & \leftarrow & MakeSpaghettiMarinara \\ MakeMeatDish & \leftarrow & MakeChickenMarinara \end{array} \qquad \begin{array}{lll} MakeSauce & \leftarrow & MakeAlfredoSauce \\ MakeSauce & \leftarrow & MakePesto \\ MakeSauce & \leftarrow & MakeMarinara \\ MakeNoodles & \leftarrow & MakeSpaghetti \\ MakeNoodles & \leftarrow & MakeFettucini \end{array}$$

The (Cooking World) Plans Knowledge Base:

$$\begin{array}{ll} MakePastaDish & \leftarrow_{\mathrm{d}} MakeNoodles \wedge MakeSauce \wedge Boil \\ MakeFettuciniAlfredo & \leftarrow_{\mathrm{d}} MakeFettucini \wedge MakeAlfredoSauce \wedge Boil \\ MakeSpaghettiPesto & \leftarrow_{\mathrm{d}} MakeSpaghetti \wedge MakePesto \wedge Boil \\ MakeSpaghettiMarinara & \leftarrow_{\mathrm{d}} MakeSpaghetti \wedge MakeMarinara \wedge Boil \\ MakeChickenMarinara & \leftarrow_{\mathrm{d}} MakeMarinara \end{array}$$

3 Analysis of Goal Recognition

As stated in Section 1, we argue that goal recognition is essentially based on a forward reasoning mechanism. To illustrate this and the other basic reasoning processes of goal recognition, let us give an example. The example is purely symbolic, i.e. with no intended interpretation for the different symbols. This should help us abstract the understanding of goal recognition.

Example Consider the Plan-KB

$$\{g_1 \leftarrow_{\mathrm{d}} s_{11} \wedge \ldots \wedge s_{1n_1}, \ldots, g_k \leftarrow_{\mathrm{d}} s_{k1} \wedge \ldots \wedge s_{kn_k}\}$$

where each s_{ij} (for $1 \leq i \leq k, 1 \leq j \leq n_i$) and g_i (for $1 \leq i \leq k$) are concepts or occurrences of concepts (i.e. with arguments).

Assume that the l^{th} observation is O_l. If O_l matches (or, more formally, unifies with) some step s_{ij}, then the observer can conclude that the agent *could be* trying to perform the i^{th} plan, that is, $s_{i1} \wedge \ldots \wedge s_{in_i}$, and hence to achieve goal g_i. This process is a forward reasoning on the Plan-KB, moving from a step in a plan to the goal that this plan achieves. This, we argue, is the central process in goal recognition.

However, it may be that observation O_l is not part of any plan, in which case the forward reasoning just explained cannot be of an immediate help. One such case is when O_l "stands for" some concept which itself appears as a step in a plan.

For instance, in the example of Section 4.1.1, the concept "Transaction" stands for "LegalTransaction" and "IllegalTransaction" as well as, transitively, for each of their subconcepts "Borrow", "Hire", etc. The "stands for" relation allows, in a goal recognition task, the refinement of an observation into some subconcepts which may themselves be parts of plans.

Example Let us consider in the logic programming framework we introduced earlier the Concept hierarchy which we represent as follows.

$$\{A \leftarrow B\ , A \leftarrow C\ , B \leftarrow B_1\ , B \leftarrow B_2\ , B \leftarrow B_3\ , B_2 \leftarrow B_4\ , B_2 \leftarrow B_5\}$$

and the Plan-KB:

$$\{g_1 \leftarrow_{\mathrm{d}} B_2 \wedge C_1 \wedge E\ , g_2 \leftarrow_{\mathrm{d}} B_5 \wedge D\ , g_3 \leftarrow_{\mathrm{d}} C_2 \wedge F\}$$

Assume that the concept B is observed. Concept B does not appear in any of the three plans that achieve goals g_1, g_2, and g_3. However, it is easy to check that, in the Concept-hierarchy, B stands for B_2 which itself is part of the plan $B_2 \wedge C_1 \wedge E$ to achieve goal g_1, and for B_5 which is part of the plan $B_5 \wedge D$ to achieve goal g_2. The process of going from observation B to concepts B_2, B_5, etc., is an abductive process. Recall that abduction is the process of concluding, from a rule $A \leftarrow B$ and an observation A, that B is a possible explanation for A.

In situations like the one we have just presented, abductive reasoning on the concept hierarchy is required to find which concepts an observation stands for. Once these concepts are found, a forward reasoning on the plans is required to find the goals that could explain the abductively produced concepts and, hence, the observation. (In the previous example, g_1 and g_2 would be plausible goals.) Note that the fact that an observation stands for some concept A that appears as a step in a plan P does not necessarily mean that plan P is necessarily intended by the agent (unless A is the only concept the observation stands for).

The above already sheds some light on the nature of goal recognition, that is, the central reasoning processes that underlie it. There are however some additional issues that need to be tackled.

Firstly, though we mentioned in Section 4.1.1 that we cannot have "negative concepts" or "negative steps in plans", which explains our choice of definite programs to represent Plan- and Concept-KBs, it is still possible to have "negative observations". By *negative observations* we mean an observation that states that a concept A (or an instance/occurrence thereof) is *definitely* not part of any plan. In other words, this is a classical negation of concept A. Clearly, it is important to relate such a negative observation to the available knowledge about plans and concepts. Two interesting cases should be considered.

1. A negative observation O unifies with a concept A with mgu θ, where $\{A \leftarrow B_1 \ , \ldots, A \leftarrow B_n\}$ is the set of clauses that define A in the concept hierarchy.

   Since we are dealing with classical negation, we can conclude—by contraposition—that each of $B_1\theta$, ..., $B_n\theta$ is also negative, that is, it cannot be observed or be part of any plan. This process can be achieved by a *backward reasoning* on the Concept-KB, which propagates the negative information backward, taking the observation as the root.

2. It may happen that each of the instances $B_1\theta$, ..., $B_n\theta$ of the concepts B_1, ..., B_n are negated, where the clauses $\{A \leftarrow B_1 \ , \ldots, A \leftarrow B_n\}$ form the definition of A in the concept hierarchy. Because of the assumption of completion of the information about A, the definition of A can be rewritten as $A \leftrightarrow B_1 \vee \ldots \vee B_n$. And since $B_1\theta$, ..., $B_n\theta$ are negated, so should $A\theta$ be. This can be thought of as a (restricted) *forward propagation of negation* on a concept hierarchy.

Finally, the reader can easily check that a backward reasoning on the Plan-KB is not useful whereas a backward propagation of negation on the Plan-KB is unsound.

So far, we have explained the processes that underlie goal recognition given some observation(s). There is still an important problem that needs attention. As we have seen, whether during the abductive or forward reasoning step, the concepts or goals that are "triggered" are not necessary but sufficient assumptions. In other words, the processes we have described try to cover all the possible goals that could be intended by the agent. Indeed, due to various complications such as ambiguities, underspecifications, etc., in the reported observations (or sentences in natural language processing), we can hardly envisage that a solution can be found to the goal recognition problem which would always return exactly the goal(s) intended by the agent. Thus, it is important to be able to select the goal(s) that is (are) most likely intended by the agent. Such a solution could be called *best* solution (in the sense of the algorithm's "best guess") to the goal recognition problem. It should satisfy the following conditions: (1) Exhaustiveness, that is, any solution should explain all the observations; (2) Consistency, that is, given any Plan and Concept knowledge bases denoted by $\mathcal{T}$, a set of observation(s) $\mathcal{O}$, and a solution $\mathcal{G}$, $\mathcal{T} \cup \mathcal{O} \cup \mathcal{G}$ should be consistent. Furthermore, on pragmatic grounds, we argue that the preferred solution should contain a number of goals smaller than or equal to that of any other solution.

4 Implementation

Our main concern in this paper is to break down the problem of goal recognition into the basic reasoning processes that underlie it. This is what we did in the previous section. Nevertheless, having done this, and having adopted a logic programming-based knowledge representation, we immediately obtain algorithms for the basic processes in goal recognition. Due to space limitations, in Section 4.1, we will only sketch the ideas behind the algorithms; the details can be found in [GM94].

4.1 An Algorithm for Goal Recognition

4.1.1 More on Knowledge Representation

As an implementation framework, we adopt the logic programming paradigm as explained in Sections 2.2 and 2.3 to represent and manipulate the Plan and Concept knowledge bases.

In fact, for efficiency purposes, we suggest that any plan of the form

$g_i \leftarrow_{\mathrm{d}} s_{i_1} \wedge \ldots \wedge s_{i_n}$

be represented as a set of facts

$\{in\_plan\_rel(g_i, s_{i_1}) \ , \ \ldots, \ in\_plan\_rel(g_i, s_{i_n})\}$

This representation is very suitable for the forward reasoning on the Plan-KB, that was presented in Section 3. It can be thought of as a table/relation *in_plan_rel* which can be accessed on two attributes, "Head" and "Step". These two attributes correspond to the first and second arguments of predicate *in_plan_rel*, respectively. This representation, which can be generated at compile time, improves the efficiency of the algorithm.

Our algorithm for goal recognition manipulates markings on concepts. It starts from the initial observation and propagates the concept markings adequately (and on logical grounds) until solutions are generated. To this end, we will use the following conventions:

- The jth observation is denoted by O_j.
- If a concept (or an instance of it) is marked as "+*j*", this means that this concept (or instance of it) matches or stands for observation O_j.
- A concept which is marked as "-1" is a concept that cannot be selected as explaining or standing for any observation. (This is what we have called a negated concept.)

In fact, since each concept (or even some instance(s) of it) may have more than one mark, we associate with each concept that is manipulated by our goal recognition algorithm a *tag* which is now defined.

Definition A *mark* is either "+*j*" or "-1", where "*j*" is an integer; the former is said to be a *positive mark* and the latter a *negative mark*.
A *label* is a pair of the form $< c', M >$ where c' is a concept or an instance of a concept c and M is a mark. A label is said to be positive (resp., negative) if the mark it contains is positive (resp., negative).
A *tag* is a list[2] $[L_1 \ldots, L_n]$, where each L_j is a label and, if $n > 1$, all the labels are positive. If the tag contains a negative label (in which case it must be unique), it will be said to be negative; otherwise positive.
A concept which has a tag (resp., label, mark) will be said to be *tagged* (resp., *labelled*, *marked*).

[2]The representation of a tag could be made less redundant by trying to factorise the instance c' of a concept name for all the labels in the list. We do not do so, however, to make the subsequent explanation easier to follow. Any implementation should deal with such details.

Note that our algorithm ensures that if a concept tag is negative, then it contains exactly one label, a negative one. This is intuitively obvious. It means that whenever a concept tag is negative, this concept cannot stand for any observation and cannot be part of any plan. So it would not make sense for it to contain some positive label.

Let us now define what it means for a concept to *cover* an observation. This will be needed for the generation of solutions in Section 4.1.5 where it will become clear why the notion of tag is needed.

Definition A concept whose tag contains a mark "+*j*" is said to *cover* observation O_j. (By definition of a tag, a concept may cover more than one observation.)

4.1.2 Abduction step

We showed in Section 3 why abductive reasoning on a concept hierarchy is needed for goal recognition. This abductive process can be achieved by a simple process of marker propagation. By passing a mark "+*j*" or "-1" from the head of a concept-KB clause to the concept in the body of this clause and iterating the process, we mark all the concepts that the observation stands for.

It turns out that, in the propositional case, a simple process of marking concepts by "+*j*" or "-1", for positive and negative observations, respectively, would have been sufficient. But since, for some applications, the knowledge bases can be non-propositional, e.g. Kautz's Operating System KB, the principle of resolution will be needed. We thus use a slight modification of SLD-resolution [Llo87], which *passes* to each resolvent a positive or negative *marking*. We call this process labelled SLD-resolution. A labelled SLD search tree is constructed for each positive or negative observation. We are mainly interested in the tags generated by these trees for the various observations. We also keep, for a particular case, the leaf nodes of such trees.

4.1.3 Forward Reasoning

The forward reasoning that was explained in Section 3 can be achieved by a propagation of marks from a "Step" attribute to a "Head" attribute of the *in_plan_rel* table presented in Section 4.1.1. Having chosen that representation of a Plan-KB, the forward propagation of marks is straightforward: it is a recursive process whereby an instance $Head\theta$ of a concept $Head$ is marked as M_k (that is, for some observation O_k), if there exists a pair *(Head,Step)* in the *in_plan_rel* table such that the tag associated with $Step$ contains the label $< Step\theta, M_k >$.

4.1.4 Negation Propagation

We are concerned here with the implementation of the forward propagation of negation which we explained in Section 3. Given that some observations and concepts are marked as "-1" to express that they are negative, the forward propagation of negation is achieved by a simple process as follows. An instance $A\theta$ of a concept A is marked as "-1" whenever $B_1\theta, \ldots B_n\theta$ are marked as "-1", where $\{A \leftarrow B_1, \ldots, A \leftarrow B_n\}$ is the definition of A in the concept knowledge base and $\theta =$ mgu$(\theta_1, \ldots, \theta_n)$.

4.1.5 Generation of Solutions

We have seen that a concept may cover more than one observation. Obviously, we are particularly interested in concepts that appear as heads of clauses in the Plan-KB since such concepts represent possible goals, i.e. solutions to the goal recognition problem. Often, no single goal covers all observations. In these cases, we must find the minimum number of goals which, put together, cover all the observations. The generalisation of this principle is the "set covering" method [RNW83]. Having a set of concepts each of which is marked (or coloured, etc.) with one or more of a predefined set of marks (colours, etc.), the problem is to find the minimal set (in terms of the number of concepts it contains) that covers all the marks (colours, etc.). We use a version of this method to generate solutions to the goal recognition problem.

We argue, on pragmatic grounds, that the less number of goals there are in any solution, the better chances one has of solving the goal recognition problem appropriately. This will be true in most situations, not all of them. However, if in any case the policy of generating the minimum number of goals à la "set-covering" is not the most appropriate, it will be the duty of any system which uses it to recover gracefully and robustly. If no solution is generated which covers all the observations, then there is an inconsistency between the observations and the knowledge bases: either the set of observations should be rejected/modified or the initial knowledge (Concept-KB and Plan-KB) is incorrect.

Evaluation of Alternative Solutions If more than one "best" solution are generated using the "set-covering" method, some additional methods could be tried. Backward reasoning can be used whenever there are more than one best solution[3]. Let $G_1, \ldots, G_n$ be those alternative solutions/goals. A simple way of selecting the best solution(s) among $G_1, \ldots, G_n$ is to evaluate each goal. In other words, for each goal G_i, check how many steps of the plan P_i (associated with G_i) have been *observed*. The goals with the highest number of observed (plan) steps should be selected for obvious pragmatic reasons. We do not believe that, for our purposes, unobserved steps provide any useful information.

If this step still does not yield one best solution, one could make use of other strategies such as "weighted abduction" [HS90, AP92] or probabilities and uncertainty-based reasoning [RZ91, CY91]. We are however not concerned here with the study of such strategies. We believe that the set-covering-based strategy which is part of our algorithm is pragmatically and practically very useful; it can also be seconded by the abductive evaluation of solutions, which we suggested above.

4.1.6 Integrated Algorithm

Having explained how the different steps of the algorithm for goal recognition can be implemented, we now give an informal statement of the integrated algorithm; details of the various steps as well as of this algorithm can be found in [GM94].

Algorithm for Goal Recognition

Input: A Concept-KB and a Plan-KB (for a given application)
A set of (positive and negative) observations
Output: A set of solutions to the goal recognition problem
The best solution (or best solutions, if more than one are equally likely).

Begin
  Using the observations, mark the corresponding nodes appropriately;
  While new marks are generated **do**
    Using *labelled SLD-resolution*, propagate the marks backward on the Concept knowledge base;
    Propagate the marks forward on the Plans knowledge base;
    propagate negative marks forward on the Concepts knowledge base;
  endwhile
  Generate the solutions using the "set-covering" method;
    If more than one "best" solution is found **then**
      Use some additional methods to find *one* best solution
    else
      Return the "best" solution and the alternative ones;
end

[3] For simplicity, and without loss of generality, let us assume that each of these solutions consists of exactly one goal.

4.1.7 Examples

Having explained the various steps of our solution to the goal recognition problem and having presented the algorithm for it, we now give some examples to illustrate the way the algorithm works. The first example is based on the (propositional) Cooking World knowledge bases; the second one is meant to show a case of an inconsistency between the observations and the knowledge bases. The last example is non–propositional. In each case, we will just state the output of each important step of the algorithm.

The Cooking World Example Consider the Concept-KB and Plan-KB presented in Section 2.3. Assume we have the following positive observations:

O_1: $MakeSpaghetti$ O_2: $MakeFettucini$ O_3: $MakeSauce$

Initially, $MarkedConcepts = [< MakeSpaghetti, +1 >, < MakeFettucini, +2 >, < MakeSauce, +3 >]$

After the labelled SLD–resolution step, we get:

$MarkedConcepts$=$[< MakeSpaghetti, +1 >, < MakeFettucini, +2 >, < MakeSauce, +3 >]$
$< MakeAlfredoSauce, +3 >, < MakePesto, +3 >, < MakeMarinara, +3 >]$

After the forward propagation of the marks on the Plan-Kb, we get[4]

$MarkedConcepts$= $MarkedConcepts \cup \{[< MakeFettuciniAlfredo, \{+2, +3\} >,$
$< MakeSpaghettiPesto, \{+1, +3\} >, < MakeSpaghettiMarinara, \{+1, +3\} >,$
$< MakePastaDish, +3 >, < MakeChickenMarinara, +3 >]\}$

No forward propagation of negative marks on Concept-KB is possible.

The set of goals that are returned are:

$Goals = [< MakeFettuciniAlfredo, \{+2, +3\} >, < MakeSpaghettiPesto, \{+1, +3\} >,$
$< MakeSpaghettiMarinara, \{+1, +3\} >, < MakeChickenMarinara, +3 >,$
$< MakePastaDish, +3 >]$

Two best solutions are returned using the set-covering method. They are:

$Best_1 = [MakeFettuciniAlfredo, MakeSpaghettiPesto]$ and

$Best_2 = [MakeFettuciniAlfredo, MakeSpaghettiMarinara]$.

Two of the other (non-best) solutions that are generated by the algorithm are:

$[MakeFettuciniAlfredo, MakeSpaghetti]$ and $[MakeSpaghettiPesto, MakeFettucini]$.

The reader can easily check that, with the same Cooking World knowledge bases, the positive observation O_1: $MakePastaDish$ and the negative observations O_2: $MakeFettucini$ and O_3: $MakeSpaghetti$, no goal (solution) covers observation O_1, the (only) positive observation. Therefore an inconsistency is signaled.

Here, we draw the reader's attention to the difference that exists between *inconsistency* as we have just illustrated it and *irrelevance* of some observation(s). The former leads to no solution covering all the observations whereas the latter simply yields multiple goals in the solution(s) generated. Observing for instance a $WashCar$ action along with the three observations of the first example in this section would merely add the goal $WashCar$ (or any goal this observation would lead to given the available knowledge) to all the solutions that are generated. $WashCar$ is thus irrelevant in the sense that it cannot be combined with, $Best_1$ or $Best_2$ (each of which covers all the three observations) to achieve a higher goal. $WashCar$ would have been "relevant" to the three observations if we had some plan like:

$$HaveParty \leftarrow_{\mathrm{d}} InviteGuests \wedge WashCar \wedge PrepareMeal \wedge ReceiveGuests$$

[4] We will use a compact representation of the tags where labels with a common concept name are grouped together. This is slightly different from the notation we presented, but it is straightforward in this (propositional) case.

The Operating System Example Consider now the following operating system taken from [Kau87]. The concept knowledge base:

```
Rename(Old,New)  ←  RenameByMove(Old,New)
Rename(Old,New)  ←  RenameByCopy(Old,New)
```

and the plans knowledge base:

```
RenameByMove(Old,New)  ←d  Move(Old,New)
RenameByCopy(Old,New)  ←d  Copy(Old,New) ∧ Delete(Old)
Modify(Old,New)        ←d  Copy(Old,New) ∧ Edit(New) ∧ Delete(Old)
```

Assume the (positive) observations are:

O_1: $Rename(a, F)$ O_2: $Rename(b, b1)$ O_3: $delete(a)$

Initially, $MarkedConcepts = [< Rename(a, F), +1 >, < Rename(b, b1), +2 >, < delete(a), +3 >]$.

By labelled SLD–resolution, the following marks are generated:

$[< RenameByMove(a, F), +1 >, < RenameByCopy(a, F), +1 >,$
$< RenameByMove(b, b1), +2 >, < RenameByCopy(b, b1), +2 >]$

The forward propagation of marks step generates the following marks:

$[< RenameByCopy(a, L), +3 >, < Modify(a, New), +3 >]$.

Since $RenameByCopy(a, F)$ and $RenameByCopy(a, L)$ unify, the reader can easily check that the algorithm generates the following best solutions:

$Best_1 = [RenameByCopy(a, F), RenameByMove(b, b1)]$ and
$Best_2 = [RenameByCopy(a, F), RenameByCopy(b, b1)]$.

Other alternative solutions could also be generated, e.g.

$[RenameByMove(a, F), Modify(a, N), RenameByMove(b, b1)]$ and
$[RenameByCopy(a, F), Modify(a, N), RenameByMove(b, b1)]$.

4.2 Incrementality

Incrementality is the problem of accommodating new observations to the current state of the goal recognition process. Our goal recognition algorithm already has an inherent notion of incrementality since it caters for several observations. In [GM94] we single out the information that should be kept for accommodating new observations as they become available.

5 Related Work

Allen [All83] analysed planning and plan recognition in terms of a set of plan construction and recognition rule pairs, reflecting plausible relationships between actions and their preconditions and effects. The application of each of these rules was governed by the use of heuristics to determine the plausibility of the consequent. In his system, plan inferencing was accomplished by a search through a set of *partial plans*, consisting of two parts. One part was constructed bottom–up using "plan inference rules" from the observed action (*The alternative*), and the other is constructed top–down using "plan construction rules" from an expected goal (*The expectation*). Partial plans could contain many expectations sharing a common alternative (That is; each observed action may have several possible explanations). Recognition was said to be reached when the two substructures merged. Allen's system was not able to deal with multiple observations.

Chaniak and McDermott [CM85] presented an algorithm based on "marker passing", which propagates marks from observations to the entire plan network. However, we failed to see, in their work, either a theoretical basis or answers to a number of questions related to the computational complexity of their solution. Thir work was enriched int a Bayessian model by Charniak and Goldman [CG93].

Closest to our work is Kautz' [Kau87, Kau90], which is regarded by many as the most formally grounded approach to the problem of plan recognition. He concentrated on *keyhole recognition* and based his work on the assumption that the observer has complete knowledge, and that the user generates *correct* plans. In his work, Kautz defines the concept of an *event hierarchy*, a set of restricted first order axioms, used to define an ontology of events and actions with two relations on the events: the *abstraction* and the *has–part* relation, respectively.

The conclusions from the plan recognition algorithm follow from the *closure* of the event hierarchy. Kautz defines the notion of C–entailment to denote that some proposition follows from the closure of the hierarchy, and relates its theory to McCarthy's system of non–monotonic inference known as *circumscription* [McC80, McC86]. Kautz makes four assumptions that together impose a circumscription on the knowledge; he also states the requirement that subsequent actions should be grouped if it is consistent to do so. The argument is then that conclusions follow deductively from the observations and the event hierarchy with the different assumptions imposed.

Kautz' algorithms are based on the manipulation of *e–graphs*, which are basically subparts of the event hierarchy that cover a set of observations. An e–graph is built from each single observation, and this structure is maintained as a representation of the plan inferred to this point. Incrementality of the plan recognition process is supported by applying the graph–matching algorithms to generate e–graphs that cover a set of observations.

Although a step towards more logic-based goal recognition systems, Kautz's model suffers from the following shortcomings. Firstly, the axioms he introduces are too procedural. For instance, his "Component-Use" assumption can be seen as a result of a "partial evaluation" of backward and forward reasoning processes. Secondly, we find his use of the "EndEvent" and "AnyEvent" concepts not intuitive or declarative. Finally, Kautz' use of e–graphs is computationally very expensive and declaratively undesirable. We believe indeed that the decision to save e–graphs to accommodate future observations is counterintuitive.

The plan recognition systems of the "second generation" (see for instance [Pol86, Pol90]) have been concerned with robustness, and, hence, the relaxation of some of the assumptions inherent in much of the previous work, e.g. that the agents construct plans that are *correct*. By its recognition of "inconsistent" plans, our model does indeed find those cases where the observations cannot be accommodated by the available knowledge. Such cases would suggest the existence of an error in the Plan-KB, the Concept-KB, or even the observations. As of now, however, our model does not make use of an attitudes model (beliefs, intentions, etc.).

In this paper, we have *not* assumed that the recogniser has any prior knowledge as to the goals of the user, unlike for instance the work of Helft and Konolige [HK90] or Pollack [Pol86, Pol90].

We finally point out that one could possibly think of an alternative implementation of our algorithms using Assumption-based Truth Maintenance Systems [Doy79, de 86]. It seems indeed that a two-layer ATMS could be used to implement the abductive reasoning on the Concept-KB, and the forward reasoning on the Plan-KB, respectively.

6 Conclusion

We presented in this paper a solution to the goal recognition problem. Our aim was to do this while paying particular attention to the nature of the problem and its underpinnings. We have represented knowledge in a logical and intuitive way, with no particular implementation in mind. Within the expressiveness of the logical formalism that we have adopted, we argued to separate the knowledge that is used in two categories: (1) conceptual knowledge which can also be thought of as an ontology of concepts that belong either to the particular application that is sought or to the background knowledge, and (2) knowledge about plans that achieve various goals, the latter being concepts that appear in the conceptual knowledge.

We then explained the various processes that take place during goal recognition, arguing that the central process in goal recognition is a forward reasoning mechanism which needs to be supported by abductive reasoning. We also presented an algorithm based on the principle of (backward or forward) marker propagation to achieve goal recognition. The examples we have given should show the reader how our algorithms work. Because of space constraints, we were not able to explain in this paper all the relevant details. The reader is referred to [GM94] for detailed algorithms, examples, and explanations.

We claim that our work is a contribution to the field in terms of the better understanding of goal recognition and also in terms of the simplicity, clarity of the solution that is presented. It could however be extended in various directions. Enhancing the semantics of the problem is one such direction. We would also like to show that our approach can handle knowledge about attitudes of dialogue partners and to enrich our framework with such knowledge. We are also interested in making use of the various solutions to the goal recognition problem that are returned by our algorithm to study the problem of cooperativity of human-machine dialogue systems.

Acknowledgements

We would like to thank Jacek Wrzos-Kaminski for interesting discussions on issues concerning the relationship between Truth-Maintenance Systems and our work and for his careful reading of and suggestions for improvements on previous drafts of this paper. We also thank Bill Black, Kristïna Jokinen, Jan Komorowski, and Ole Jakob Mengshoel, as well as the anonymous referees for their useful comments on an earlier version of the paper. We are also grateful to the members of the Knowledge Systems Group in our department for constant encouragements and the challenging scientific environment they have provided us with.

References

[All83] J. Allen. Recognizing intentions from natural language utterances. In M. Brady and R. C. Bervick, editors, *Computational Models of Discourse, Chapter 12.* Bradford, MIT Press, Cambridge, 1983.

[AP92] D.E. Appelt and M.E. Pollack. Weighted abduction for plan ascription. *User Modeling and User-adapted Interaction*, 2:1–25, 1992.

[CG93] E. Charniak and R.P. Goldman. A Bayesian Model of Plan Recognition. *Artificial Intelligence*, 64:53–79, 1993.

[CM85] E. Charniak and D. McDermott. *Introduction to Artificial Intelligence.* Addison Wesley, 1985.

[CY91] R.J. Calistri-Yeh. Utilizing user models to handle ambiguity in robust plan recognition. *User Modeling and User-adapted Interaction*, 1(4):289–322, 1991.

[de 86] J. de Kleer. An Assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, 1986.

[Doy79] J. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12:231–272, 1979.

[Eis79] C. Eisele. *Studies in the Scientific and Mathematical Philosophy of Charles S. Peirce.* Mouton Publishers, 1979.

[Fan70] K.T. Fann. *Peirce's theory of abduction.* Martinus Nijhoff, The Hague, 1970.

[FN71] Fikes and Nilsson. Strips: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2:189–208, 1971.

[GM89] G. Gazdar and C. Mellish. *Natural Language Processing in Prolog.* Addison Wesley Publishing Company, 1989.

[GM94] A. Guessoum and T. Mollestad. Towards a Logical Reconstruction of Plan Recognition. Technical report, Dept. of Computer Systems and Telematics, The Norwegian Institute of Technology, March 1994, revised April 1994.

[HK90] Helft and Konolige. Plan Recognition as Abduction and Relevance (Draft version), Artificial intelligence center, SRI international, Menlo Park, California. 1990.

[HS90] Hobbs and Stickel. Interpretation as Abduction. Technical report, SRI International, 1990.

[Kau87] H.A. Kautz. A formal Theory of Plan recognition, may 1987. Technical report, University of Rochester, NY, 1987.

[Kau90] H. Kautz. A circumscriptive theory of plan recognition. In P.R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication.* Bradford, MIT Press, Cambridge, 1990.

[Kon94] K. Konolige. Plan Recognition as Abduction and Relevance. Technical report, SRI International, 1994.

[LA87] D.J. Litman and J. Allen. A Plan Recognition Model for Subdialogues in Conversations. *Cognitive Science*, 11:163–200, 1987.

[Lit85] D. Litman. *Plan Recognition and Discourse Analysis.* PhD thesis, University of Rochester, Rochester, USA, 1985.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming.* Springer Verlag, 1987.

[McC80] J. McCarthy. Circumscription – A form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McC86] J. McCarthy. Applications of Circumscription to Formalizing Common Sense Knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[MH69] J. McCarthy and P.J Hayes. Some Philosophical Points from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1969.

[Mol89] T. Mollestad. *Natural Language Interfaces to Databases – Cooperating with the Casual User.* Unpublished MSc Thesis, University of Trondheim NTH, 1989.

[Pol86] M. E. Pollack. A Model of Plan Inference that distinguishes between the Beliefs of Actors and Observers. In P. Georgeff, M and A. Lansky, editors, *the 1986 Workshop: Reasoning about Actions and Plans*, pages 279–295, 1986.

[Pol90] M. E. Pollack. Plans as Complex Mental Attitudes. In P.R. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication.* Bradford, MIT Press, Cambridge, 1990.

[RNW83] J.A. Reggia, D.S. Nau, and P.Y. Wang. Diagnostic Expert Systems based on a Set Covering Model. *Int. Journal of Man–Machine Studies*, 19:437–460, 1983.

[RZ91] B. Raskutti and I. Zukerman. Generation and selection of likely interpretations during plan recognition. *User Modeling and User-adapted Interaction*, 1(4):322–353, 1991.

[Sac75] E.D. Sacerdoti. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence*, 5(2):115–135, 1975.

[Sid83] C.L. Sidner. What the Speaker Means: The Recognition of Speaker's Plans in Discourse. *Computers and Mathematics with Applications*, 9(1):71–82, 1983.

[Wil83] R Wilensky. *Planning and Understanding: A computational approach to Human Reasoning.* Addison Wesley Publishing Company, 1983.