Plan Recognition through Goal Graph Analysis

Jun Hong[1]

Abstract. We present a novel approach to plan recognition based on a two-stage paradigm of graph construction and analysis. First, a graph structure called a Goal Graph is constructed to represent the observed actions, the state of the world, and the achieved goals as well various connections between these nodes at consecutive time steps. Then, the Goal Graph is analysed at each time step to recognise those achieved goals consistent with the actions observed so far and the valid plans for the recognised goals or part of the recognised goals. We describe two algorithms for Goal Graph construction and analysis in this paradigm, that are both provably sound, polynomial-time and polynomial-space. We have tested these algorithms in two domains with up to 245 goal schemata and 100000 possible goals, in which excellent performance has been achieved in terms of efficiency, accuracy and scalability.

1 Introduction

Plan recognition involves inferring the goal of an agent from a set of observed actions and organising the observed actions into a plan structure for the goal. Most plan recognition systems (e.g., [4]) search a space of possible plans for candidate plans that account for the observations. To form the search space, some kind of plan representation is required. For instance, in Kautz's event hierarchy, plan decompositions are required that describe how low level actions make up complex actions. Despite its obvious advantage of expressive richness, plan representation has a limitation in its inability to deal with new plans whose types do not appear in the plan representation. Hand-coding the plan representation in a large and complex domain presents a tedious or impractical task. In some other domains, the knowledge about plans might not be readily available. Some attempts (e.g., [7], [6], [1]) have recently been made to use machine learning techniques to automate acquisition of plan representation. Even when leaving aside the plan representation consideration, searching the plan space can be exponentially expensive because the number of possible plans can be exponential in the number of actions [4]. Most plan recognition systems have often been developed in domains in which there are fewer than 100 plans and goals [6].

We present a novel approach to plan recognition, in which graph construction and analysis is used as a paradigm. This approach significantly differs from most plan recognition systems. Instead of searching for a plan as in these systems, a graph structure called a Goal Graph is first constructed to represent the observed actions, the state of the world as it is changed by these actions, and the fully or partially achieved goals at consecutive time steps. Connections are also made between different kinds of nodes in the Goal Graph. The constructed Goal Graph can then be analysed at each time step to recognise those fully or partially achieved goals that are consistent with the actions observed so far. The Goal Graph analysis also reveals causal links over actions and goals so that valid plans for the recognised goals or part of the recognised goals can be further recognised. We only define what constitutes a valid plan for a goal eliminating the need for plan representation. Under our formulation, the plan recogniser must consider how the observed actions can be composed into plans. Our formation is not limited in its ability to recognise new plans.

We describe two algorithms, GoalGraphConstructor and GoalGraphAnalyser, based on this paradigm. We prove that our algorithms are sound, polynomial-time and polynomial-space. The algorithms have been tested on a 500 MHz Pentium III in two domains. In the extended briefcase domain, we increase the number of locations and objects to create a series of sets of up to over 100000 possible goals for testing the scalability of our algorithms where the approximate linear time performance has been achieved. In the Unix domain, we use a set of data collected in the Unix domain at the University of Washington with over 245 goal schemata and over 10000 possible goals. In this domain, on average it only takes less than a CPU second to update the Goal Graph when an observed action is processed and usually only a very small number of consistent goals remain after a sequence of observed actions has been processed.

2 The Domain Representation

We use an ADL-like representation [9], including actions with conditional and universally quantified effects, and existentially as well as universally quantified preconditions and goal descriptions. A plan recognition problem consists of

- A set of action schemata specifying primitive actions.
- A finite, dynamic universe of typed objects.
- A set of propositions called the Initial Conditions.
- A set of goal schemata specifying possible goals.
- A set of observed actions that are partially ordered.
- An explicit notion of discrete time.

The solution to a plan recognition problem consists of a set of recognised goals that are consistent with the observed actions together with the valid plans consisting of the observed actions for the recognised goals or part of the recognised goals.

The goal schema consists of a set of goal descriptions. The action schema consists of a set of preconditions and a set of effects. The set of goal descriptions for a goal must be satisfied in the state of the world when the goal is fully achieved. If some but not all goal descriptions are satisfied instead, the goal is partially achieved. The set of preconditions must be satisfied in the state of the world before an action can be executed. The set of effects are taken in the state of the world when an action is executed. In the actual implementation

[1] School of Information and Software Engineering, University of Ulster at Jordanstown, Newtownabbey, Co. Antrim BT37 0QB, UK, email: j.hong@ulst.ac.uk

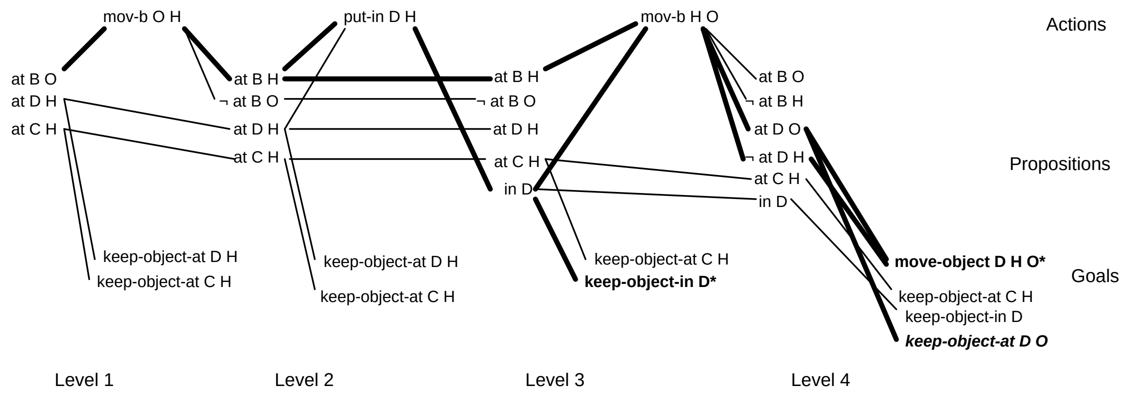

Figure 1. A Goal Graph for an example of the extended briefcase domain

of our plan recognition algorithms, universally quantified preconditions and effects, and conditional effects in an action schema as well as universally quantified goal descriptions in a goal schema are eliminated and equivalent schemata are created.

3 Constructing a Goal Graph

3.1 Goal Graphs

A Goal Graph is a directed, levelled graph. The levels alternate between proposition levels containing proposition nodes (each labelled with a proposition or negation of a proposition) representing the propositions true or explicitly known to be false in the state of the world at consecutive time steps, goal levels containing goal nodes (each labelled with a goal) representing goals fully or partially achieved at consecutive time steps, and action levels containing action nodes (each labelled with an action) representing actions observed at consecutive time steps. The levels in a Goal Graph start with a proposition level at time step 1 that consists of one node for each proposition true in the Initial Conditions. They end with a goal level at the last time step that consists of a node for each of the goals fully or partially achieved so far.

The goal nodes in goal-level i are connected by description edges to their goal descriptions in proposition-level i. The action node in action-level i is connected by precondition edges to its preconditions in proposition-level i, and by effect edges to its effects in proposition-level $i+1$. Those proposition nodes in proposition-level i are connected by persistence edges to the corresponding proposition nodes in proposition-level $i+1$ if their truth values have not been affected by the effects of the action in action-level i.

Figure 1 shows a Goal Graph for an example of the extended briefcase domain [8], which involves transportation of two physical objects, a dictionary and a chequebook, between home and office. Three actions have been observed at three consecutive time steps: `(mov-b O H)`, `(put-in D H)`, and `(mov-b H O)`. The Initial Conditions consist of: `(at B O)`, `(at D H)` and `(at C H)`. Action and goal nodes are on the top and bottom parts of the graph respectively. The proposition nodes are in the middle part of the graph.

3.2 Goal Graph Construction Algorithm

The GoalGraphConstructor takes a set of partially ordered actions as they are observed and constructs a Goal Graph. We use a 4-tuple, $< P, A_O, G_R, E >$, to represent a Goal Graph, where P is a set of proposition nodes, A_O is a set of action nodes, G_R is a set of goal nodes, and E a set of edges. The algorithm starts with a Goal Graph, $< P, \{\}, \{\}, \{\} >$, that consists of only proposition-level 1 with nodes representing the Initial Conditions.

Given a Goal Graph ending with proposition-level i, the GoalGraphConstructor first extends the Goal Graph to goal-level i with nodes representing goals fully or partially achieved at time step i. Meanwhile, if a node in proposition-level i satisfies a goal description, a description edge connecting the proposition node to the goal node is added onto the Goal Graph. Figure 2 shows the goal expansion algorithm. The algorithm takes a Goal Graph $< P, A_O, G_R, E >$, time step i, and a set of goal schemata G as input and returns an updated Goal Graph after the goal expansion.

```
Goal-Expansion($< P, A_O, G_R, E >, i, G$)
1. For every $G_k \in G$
     For every instance $g$ of $G_k$
       a. Get a set of goal descriptions $S_g$.
       b. Get the equivalent set of $S_g$, $S_{g'}$.
       c. For every $p_g \in S_{g'}$, where $p_g = not(p_{g'})$,
            If $prop(neg(p_{g'}), i) \in P$, then
              Add description-edge($prop(neg(p_{g'}), i)$, $goal(g, i)$) to $E$.
       d. For every $p_g \in S_{g'}$, where $p_g \neq not(p_{g'})$,
            If $prop(p_g, i) \in P$, then
              Add description-edge($prop(p_g, i)$, $goal(g, i)$) to $E$.
       e. If one of the goal descriptions of $g$ is satisfied, then
            Add $goal(g, i)$ to $G_R$.
2. Return with $< P, A_O, G_R, E >$.
```

Figure 2. The goal expansion algorithm

When an action is observed at time step i, the GoalGraphConstructor then extends the Goal Graph ending with goal-level i, to action-level i with a node representing the observed action. At the same time, the algorithm also extends the Goal Graph to proposition-level $i+1$ with nodes representing propositions true or explicitly known to be false after the action has been observed. Meanwhile, if a node in proposition-level i satisfies a precondition of the action, a precondition edge connecting the proposition node to the action node is added onto the Goal Graph. For every effect of the action, the GoalGraphConstructor simply adds a proposition node representing the effect to proposition-level $i+1$. The effect edge from the action node to the proposition node is also added onto the Goal Graph. All the propositions nodes at proposition-level i are brought forward to proposition-level $i+1$ by maintenance actions if their truth values have not been changed by the effects of the action observed at time step i. Persistence edges connecting the proposition nodes at the two proposition levels are added onto the Goal Graph. Figure 3 shows the action expansion algorithm. The algorithm takes a Goal Graph $< P, A_O, G_R, E >$, the observed action a_i, time step i, and a set of action schemata A as input and returns an updated Goal Graph after the action expansion.

```
Action-Expansion($< P, A_O, G_R, E >$, $a_i$, $i$, $A$)
1. Add $action(a_i, i)$ to $A_O$.
2. Instantiate an action schema in $A$ with $a_i$ to get a set of
   preconditions $S_P$, and a set of effects $S_E$.
3. Get the equivalent sets of $S_P$ and $S_E$, $S_{P'}$ and $S_{E'}$.
4. For every $p_p \in S_{P'}$, where $p_p = not(p_{p'})$,
     If $prop(neg(p_{p'}, i) \in P$, then
      Add precondition-edge($prop(neg(p_{p'}, i), action(a_i, i))$ to $E$.
5. For every $p_p \in S_{P'}$, where $p_p \neq not(p_{p'})$,
     If $prop(p_p, i) \in P$, then
      Add precondition-edge($prop(p_p, i), action(a_i, i))$ to $E$.
6. For every $p_e \in S_E$
     a. Add $prop(p_e, i+1)$ to $P$.
     b. Add effect-edge($action(a_i, prop(p_e, i+1))$ to $E$.
7. For every $prop(p, i) \in P$
     If $prop(\neg p, i+1) \notin P$, then
      If $prop(p, i+1) \notin P$, then Add $prop(p, i+1)$ to $P$;
      Add persistence-edge($prop(p, i), prop(p, i+1))$ to $E$.
8. Return with $< P, A_O, G_R, E >$.
```

Figure 3. The action expansion algorithm

Theorem 1 (Polynomial Size and Time) *Consider a plan recognition problem with t observed actions in t time steps, a finite number of objects at each time step, p propositions in the Initial Conditions, and m goal schemata each having a constant number of parameters. Let l_1 be the largest number of the effects of any of the action schemata, l_2 be the largest number of the goal descriptions of any of goal schemata. Let n be the largest number of objects at all time steps. Then, the size of the Goal Graph of $t+1$ levels created by the GoalGraphConstructor, and the time needed to create the graph, are polynomial in n, m, p, l_1, l_2 and t.*

The maximum number of nodes in any proposition level is $O(p + l_1 t)$. Let k be the largest number of parameters in any goal schema. Since any goal schema can be instantiated in at most n^k distinct ways, the maximum numbers of nodes and edges in any goal level are $O(mn^k)$ and $O(l_2 mn^k)$ respectively. It is obvious that the time needed to create both nodes and edges in any level is polynomial in the number of nodes and edges in the level.

Theorem 2 *The GoalGraphConstructor is sound: Any goal it adds to the Goal Graph at time step i is one either fully or partially achieved at time step i in the state of the world. The algorithm is complete: If a goal has been either fully or partially achieved by the observed actions up to time step $i-1$, then the algorithm will add it to the Goal Graph at time step i under the assumption that all possible goals are restricted to the categories of goal schemata.*

Proposition-level i of the Goal Graph represents the state of the world at time step i that has been changed from the Initial Conditions after the actions have been observed at time step 1, ..., $i-1$. On the other hand, goal-level i of the Goal Graph consists of all possible instances of the goal schemata that are fully or partially achieved in the state of the world at time step i .

4 Recognising Consistent Goals and Valid Plans

4.1 Valid Plans

We now define what mean when we say a set of partially ordered actions forms a valid plan for a goal given the Initial Conditions.

Definition 1 (Causal Link) *Let a_1 and a_2 be two actions. There exists a causal link between a_1 and a_2, written as $a_1 \rightarrow a_2$, if and only if one of the effects of a_1 satisfies one of the preconditions of a_2.*

A goal can be treated as an action with goal descriptions as its preconditions and an empty set of effects. Therefore causal links can also be established from actions to goals.

Definition 2 (Valid Plan) *Let g be a goal, and $P =< A, O, L >$ where A is a set of actions, O is a set of temporal ordering constraints, $\{a_i < a_j\}$, over A, and L is a set of causal links, $\{a_i \rightarrow a_j\}$, over A. Let I be the Initial Conditions. P is a valid plan for g, given I, if and only if*

1. *the actions in A can be executed in I in any order consistent with O;*
2. *the goal g is fully achieved after the actions in A are executed in I in that order.*

4.2 Consistent Goals

We finally define what we mean when we say a goal is consistent with a set of partially ordered actions.

Definition 3 (Relevant Action) *Given a goal g and a set of partially ordered actions, $< A, O >$, where A is a set of actions, O is a set of temporal ordering constraints, $\{a_i < a_j\}$, over A, an action $a \in A$ is said to be relevant to g in the context of $< A, O >$, if and only if*

1. *there exists a causal link, $a \rightarrow g$; or*
2. *there exists a causal link, $a \rightarrow b$, where $b \in A$ is relevant to g and $a < b$ is consistent with O.*

Definition 4 (Consistent Goal) *A goal g is consistent with a set of partially ordered actions, $< A, O >$, if and only if every $a \in A$ is relevant to g in the context of $< A, O >$.*

Proposition 1 (Valid Plan for Consistent Goal) *Let $< A, O >$ be a set of partially ordered actions, I be the Initial Conditions before $< A, O >$, g be a goal consistent with $< A, O >$. Given I, $P =< A, O, L >$, where L is a set of causal links $\{a_i \rightarrow a_j\}$ over A, is a valid plan for either g when g is fully achieved after $< A, O >$ or the achieved part of g when g is partially achieved after $< A, O >$.*

Proposition 1 follows Definition 2, 3 and 4. Especially when g is partially achieved, let $g\prime$ be the achieved part of g. So $g\prime$ is fully achieved and $P =< A, O, L >$ is a valid plan for $g\prime$.

4.3 Goal Graph Analysis Algorithm

The GoalGraphAnalyser analyses the constructed Goal Graph to recognise consistent goals and valid plans. We assume that every observed action is relevant to the goal intended by the agent in the context of the agent's actions. Therefore, the goal intended by the agent is consistent with the observed actions and a goal may be the intended goal if it is consistent with the set of the observed actions. Theorem 3 and Theorem 4 state how the recognition of the consistent goals can be achieved by the analysis of a constructed Goal Graph.

Theorem 3 *Given a Goal Graph, there exists a causal link, $a_i \rightarrow g_j$ between an action a_i at time step i and a goal g_j at time step j, where $i < j$, if a_i is connected to g_j via a path of an effect edge, zero or more persistence edges and a description edge. We call such a path a causal link path between a_i and g_j.*

Theorem 4 *Given a Goal Graph, there exists a causal link, $a_i \rightarrow a_j$, and a temporal ordering constraint, $a_i < a_j$, between an action a_i at time step i and another action a_j at time step j, where $i < j$, if a_i is connected to a_j via a path of an effect-edge, a zero or more persistence-edges and a precondition-edge. We call such a path a causal link path between a_i and a_j.*

Based on the structure of the Goal Graph, we can prove the existence of the causal link, $a_i \rightarrow g_j$, in Theorem 3 and $a_i \rightarrow a_j$, in Theorem 4. It is obvious that a causal link path between a_i and a_j guarantees the temporal ordering constraint, $a_i < a_j$.

Given a constructed Goal Graph $< P, A_O, G_R, E >$ of k levels, the GoalGraphAnalyser shown in Figure 4 recognises every consistent goal from the goals in goal-level k by deciding whether every observed action is relevant to it. This is done by first finding those relevant actions from the observed actions, that are connected to the goal by causal link paths. For each of the already-known relevant actions, the algorithm tries to find more relevant actions from the observed actions, that are connected to it by causal link paths. This continues until no more relevant action is found. The consistent goal recognised and the valid plan for the goal or part of it are represented by a 3-tuple, $< g_k, < A_O, O, L_a >, L_g >$, where g_k is the goal, L_a is a set of causal links over the observed actions, and L_g is a set of causal links between some of the observed actions and the goal. $< A_O, O, L_a >$ represents a valid plan for g_k and L_g further explains the purposes of some of the observed actions.

Theorem 5 *The GoalGraphAnalyser is sound: Any goal g it recognises at time step t is consistent with the observed actions so far, and the plan it organises for g or part of g is valid.*

In the example shown in Figure 1, the goal nodes in bold represent three consistent goals among which the goal node in italics represents a partially achieved goal while the other two represent two fully achieved goals. The edges in bold show causal link paths.

Theorem 6 (Polynomial Space and Time) *Consider a t-level Goal Graph. Let l_1 be the number of fully or partially achieved goals at time step t, m_1 be the largest number of goal descriptions in any of these goals, l_2 be the number of the observed actions, and m_2 be the largest number of preconditions in any of these actions. The space size of possible causal link paths that connect the goals to the observed actions and that connect the observed actions to other observed actions, and the time needed to recognise all the consistent goals are polynomial in l_1, l_2, m_1 and m_2.*

```
GoalGraphAnalyser($< P, A_O, G_R, E >, k$)
1. For every $g_k \in G_R$ in goal-level $k$
    a. $A_O\prime \leftarrow \{\}, A \leftarrow \{\}, L_g \leftarrow \{\}, L_a \leftarrow \{\}$.
    b. For every $a_i \in A_O$ connected to $g_k$ by a causal link path
        Add $a_i \rightarrow g_k$ to $L_g$.
        Add $a_i$ to $A_O\prime$.
        Add $a_i$ to $A$.
    c. If $A = \{\}$ and $A_O = A_O\prime$, then
         Get a set of all temporal ordering constraints over $A_O$, $O$.
         Add $< g_k, < A_O, O, L_a >, L_g >$ to GoalPlan, else
    d. Remove an action $a_j$ from $A$.
    e. For every $a_i \in A_O$ connected to $a_j$ by a causal link path
        Add $a_i \rightarrow a_j$ to $L_a$.
        If $a_i \notin A_O\prime$, then
         Add $a_i$ to $A_O\prime$.
         Add $a_i$ to $A$.
         Go to 1c.
2. Return with GoalPlan.
```

Figure 4. The Goal Graph analysis algorithm

Persistence edges do not branch in a Goal Graph. For each of the goals in goal-level t, the maximum number of paths searched for those observed actions, that are connected to it by causal link paths and hence relevant to it, is $O(m_1)$. For each of the relevant actions to the goal, the maximum number of paths searched for those observed actions, that are connected to it by causal link paths and hence also relevant to the goal, is $O(m_2)$. There are only at maximum l_1 goals in goal-level t and l_2 relevant actions to any of these goals. So the space size of possible causal link paths is $O(l_1(m_1 + l_2 m_2))$. The time needed to recognise all the consistent goals is polynomial in the space size.

5 Experimental Results

Our algorithms have been implemented in Prolog and tested on a 500 MHz Pentium III in two domains in terms of efficiency, accuracy and scalability. In the extended briefcase domain, we increase the number of locations to 50 and the number of objects up to 40 to create a series of spaces of 10,000, 20,000, up to 100,000 possible goals respectively. The same sequences of observed actions with the same Initial Conditions are used in the experiments in conjunction with these spaces of possible goals. Figure 5 shows that the average CPU time taken to process an observed action is approximately linear in the number of goals.

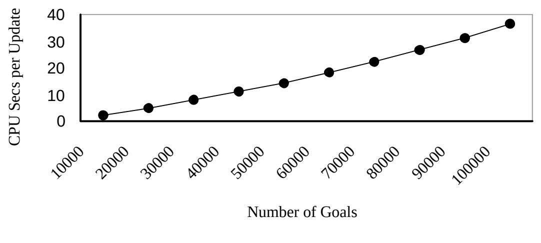

Figure 5. Experimental results of the extended briefcase domain

In the Unix domain, we tested our algorithms on a set of data collected at the University of Washington with regard to efficiency and accuracy. To collect the data, the subjects are given goals described in English first and they then try to solve each goal by executing Unix commands. The executed Unix commands are recorded in the data set. We have 29 action schemata for the Unix commands, 245 goal schemata and an estimate of 10000 possible goals. The results show that on average it only takes less than a CPU second to process an observed action and usually only a very small number of consistent goals remain after a sequence of observed actions have been processed.

Table 1. Experimental results of the Unix domain

goal	cpu sec per update	length of observation	fully achieved goals	partially achieved goals	remaining goals
G_1	1.202	2.25	7	12	4
G_2	0.352	16	10	9	1
G_3	0.027	3.0	3	10	1
G_4	1.255	20.5	33	0	1

Table 1 gives a summary of the experimental results. We tested our algorithms on four goals that were originally tested in [5]. The CPU second per update is the average time it takes to process an observed action. The length of observation is the average number of observed actions executed by the subjects to achieve the given goal. The fully achieved goals are the goals fully achieved after the last action has been observed. The partially achieved goals are the goals partially achieved. The remaining goals are the goals recognised after the last observed goal has been processed. These results show that our algorithms perform extremely well with regard to efficiency and accuracy. They also demonstrate a significant improvement on the performance of the goal recogniser [5] where 155, 37 and 15 goals remain on G_1, G_2 and G_4 respectively.

6 Related Work

Blum and Furst [2], [3] introduced a new graph-based approach to planning in STRIPS domains, in which a graph structure called a Planning Graph is first constructed explicitly rather than searching immediately for a plan as in standard planning methods. The Planning Graph is then analysed to generate possible plans. Our Goal Graph-based approach to plan recognition can be seen as a counterpart of planning with Planning Graph. Though graph structures are used in both approaches, they consist of different kinds of nodes and edges, take different inputs and aim at producing different outputs. The domain representation can be the same for both planning with Planning Graph and plan recognition with Goal Graph. In this regard, the previous efforts on handling more expressive representation languages (see [10] for details) are still useful for plan recognition.

Lesh and Etzioni tried to use a graph representation of actions and goals for the goal recognition problem [5]. Their graph representation only consists of action and goal nodes that are fully connected to each other first and then inconsistent goals are repeatedly pruned from this graph representation. This will lead to a set of candidate goals that explain the observed actions. Their graph representation does not explicitly represent temporal ordering constraints and causal links over actions and goals. So their system can only recognise goals rather than plans because it cannot organise the observed actions into plan structures for the recognised goals. Their system is sound and polynomial-time. They have however sacrificed expressiveness of plan representation for tractability. This is not the case in our system that recognises both goals as well as plans and performs in polynomial time and space. As we indicated in the previous section the number of remaining goals after pruning in their system is usually large.

7 Conclusion

In this paper, we introduced a new approach to plan recognition in which a graph structure called a Goal Graph is constructed and analysed for plan recognition. We described two algorithms for constructing and analysing a Goal Graph. Our algorithms recognise both goals and plans. They allow redundant and partially ordered actions. They are sound, polynomial-time and polynomial-space.

Our empirical experiments show that our algorithms are computationally efficient and they can be scaled up and applied to domains where there are tens of thousands of goals and plans. They recognise goals and plans with great accuracy. Our approach has therefore accommodated both expressiveness and tractability without the use of plan representation. Since our new graph-based approach to plan recognition is fundamentally different from the existing methods for plan recognition, it provides an alternative to these methods and shows a new perspective of research into plan recognition.

Our plan recognition system is limited in its ability to recognise every type of erroneous plans, e.g., an erroneous plan involving an observed action completely irrelevant to the intended goal. The GoalGraphAnalyser is not complete: it may not immediately recognise the intended goal as a consistent one when the action currently observed has a causal link with a relevant action that has not yet been observed. So it may temporarily miss the intended goal if it is not yet in the set of consistent goals. This is inevitable to the human observer: when an observed action is not yet found relevant to a goal in a consistent way, we can either make an unsound guess that it could be relevant to some of the consistent goals we have at moment, or delay the decision for a little while until more actions are observed and this currently observed action is found relevant to a goal in a consistent way. Despite these limitations, our system performs extremely well in our two test domains.

REFERENCES

[1] M. Bauer, ‘Acquisition of abstract plan descriptions for plan recognition’, in *Proceedings of AAAI-98*, pp. 936–941, Madison, Wisconsin, (1998).

[2] A.L. Blum and M.L. Furst, ‘Fast planning through planning graph analysis’, in *Proc. of IJCAI-95*, pp. 1636–1642, Montreal, (1995).

[3] A.L. Blum and M.L. Furst, ‘Fast planning through planning graph analysis’, *Artificial Intelligence*, **90**, 281–300, (1997).

[4] H.A. Kautz, *A Formal Theory of Plan Recognition*, PhD Thesis, University of Rochester, 1987.

[5] N. Lesh and O. Etzioni, ‘A sound and fast goal recognizer’, in *Proc. 15th Int. Joint Conf. on AI*, pp. 1704–1710, (1995).

[6] N. Lesh and O. Etzioni, ‘Scaling up goal recognition’, in *Proc. of Int. Conf. on Principles of Knowledge Representation and Reasoning*, (1996).

[7] R.J. Mooney, ‘Learning plan schemata from observation: explanation-based learning for plan recognition’, *Cognitive Science*, 483–509, (1990).

[8] E. Pednault, ‘Synthesizing plans that contain actions with context-dependent effects’, *Computational Intelligence*, **4**(4), 356–372, (1988).

[9] E. Pednault, ‘Adl: Exploring the middle ground between strips and the situation calculus’, in *Proc. of KR-89*, pp. 324–332. Morgan Kaufman, (1989).

[10] Daniel S. Weld, ‘Recent advances in ai planning’, *AI Magazine*, **20**(2), 93–123, (Summer, 1999).