Plan and Goal Recognition as HTN Planning

Daniel Höller and **Pascal Bercher** and **Gregor Behnke** and **Susanne Biundo**
Institute of Artificial Intelligence, Ulm University, D-89069 Ulm, Germany
{daniel.hoeller, pascal.bercher, gregor.behnke, susanne.biundo}@uni-ulm.de

Abstract

Plan- and Goal Recognition (PGR) is the task of inferring the goals and plans of an agent based on its actions. A few years ago, an approach has been introduced that successfully exploits the performance of planning systems to solve it. That way, no specialized solvers are needed and PGR benefits from present and future research in planning. The approach uses *classical* planning systems and needs to plan (at least) once for *every possible goal*. However, models in PGR are often structured in a hierarchical way, similar to Hierarchical Task Networks (HTNs). These models are strictly more expressive than those in classical planning and can describe partially ordered sets of tasks or multiple goals with interleaving plans. We present the approach *PGR as HTN Planning* that enables the recognition of complex agent behavior by using unmodified, off-the-shelf HTN planners. Planning is thereby needed only *once*, regardless of how many possible goals there are. Our evaluation shows that current planning systems are able to handle large models with thousands of possible goals and that the approach results in high recognition rates.

1 Introduction

For systems that interact with other agents, it may be important to know which goals their counterparts want to achieve and what actions might be performed next. This information may be useful to planning its own behavior, to decide when to trigger own behavior (e.g. when to offer help to a user), or to prior the activity recognition, i.e., the component recognizing the performed actions. The task is commonly known as *Plan and Goal Recognition* (PGR) and it is usually solved based on a sequence of actions that the observed agent performed. There have been different approaches to solve it, e.g., rule-based, based on abduction, probabilistic inference, or techniques from parsing (Sukthankar et al. 2014).

A few years ago, an approach has been introduced that exploits the performance of classical planning systems to solve it. The earliest work in this sub-field was presented by Ramírez and Geffner, in a first step by using modified planning systems (2009), later by using off-the-shelf systems (2010). By using unchanged systems, such an approach also benefits from ongoing research in planning, without any adaptation. Their approach runs the planner twice for *each possible goal*, once by enforcing any solution to start with the given prefix, and once planning from scratch. By comparing the costs of the found solutions, the system estimates how likely the goal is. Recently, there has been work to extend the approach to support features like observations over fluents (instead of *actions*) or uncertainty in the observations (Sohrabi, Riabov, and Udrea 2016).

Since the models in PGR express the behavior of one or more observed agents, using models from automated planning seems to be a good choice. However, using STRIPS means to restrict the models to a basic level of expressivity (that equals that of regular languages (Höller et al. 2016)). Models in PGR often use hierarchical (grammar-like) structures (e.g. Geib and Goldman 2011). These are also common in planning: the most widely used (hierarchical) formalism is Hierarchical Task Network (HTN) planning (Erol, Hendler, and Nau 1996). HTN models can express (non-context-free) context-sensitive structures (Höller et al. 2014), and have also been proposed to be used in PGR (Geib 2004), though its expressivity makes PGR, in the general case, undecidable (Behnke, Höller, and Biundo 2015). The hierarchy enables a natural definition of the agents' goals as the top-most tasks. From a modeling perspective, HTN models combine STRIPS-like state-based definitions with expressive grammar structures. These may even include partially ordered tasks. This is especially useful to recognize several goals of an agent with interleaving plans, or the goals of several observed agents.

We present the approach *PGR as HTN Planning* that enables the recognition of complex agent behavior by using unmodified, off-the-shelf HTN planners. The behavior of the observed agent(s) is described by arbitrary HTN definitions. We present a transformation that restricts the set of solutions to an HTN planning problem to those starting with a given prefix of observed actions. Our overall approach allows us to plan only once, regardless of how many possible goals there are. Our empirical evaluation shows that the approach works well on a large plan and goal recognition corpus that includes nearly 2000 distinct goals the agent may pursue.

Section 2 introduces the formal framework and defines the PGR problem. Section 3 introduces the approach that is evaluated in Section 4. We finally discuss the approach in Section 5.

2 Planning Framework

This section introduces the HTN formalism of Geier and Bercher (2011) and defines the PGR problem in this context.

2.1 Hierarchical Planning

In HTN planning, there are two types of tasks, *compound* (or *abstract*) and *primitive* tasks (or *actions*). Abstract tasks are successively decomposed into other tasks (that may be abstract or primitive) until only primitive tasks are left. These can directly be executed and are similar to actions in STRIPS.

The sets of primitive task names (actions) and compound task names are denoted as A and C, respectively. Tasks are organized in *task networks*. A task network tn is a triple $(T, \prec, \alpha)$. T is the (possibly empty) set of unique task *identifiers*. These are mapped to task *names* by the function $\alpha : T \to C \cup A$. This enables a single task name (e.g. *move-a-b*) to appear multiple times in a task network. $\prec \subseteq T \times T$ defines a partial order on the task identifiers. Two task networks $tn = (T, \prec, \alpha)$ and $tn' = (T', \prec', \alpha')$ are called *isomorphic* ($tn \cong tn'$) if they differ solely in their identifiers, i.e., if there is a bijection $\sigma : T \to T'$ so that for all $t, t' \in T$ holds that $[(t, t') \in \prec] \Leftrightarrow [(\sigma(t), \sigma(t')) \in \prec']$ and $\alpha(t) = \alpha'(\sigma(t))$.

Abstract tasks are decomposed by using *(decomposition) methods*. Let M be the set of all methods. A method $m \in M$ is a pair (c, tn) that maps an abstract task $c \in C$ to a task network tn, which specifies the (primitive or abstract) *subtasks* of c and their ordering. When a task is decomposed, it is deleted from the network, the subtasks specified in the method are inserted and inherit the ordering relations from the deleted task. Formally, a method (c, tn) decomposes a task network $tn_1 = (T_1, \prec_1, \alpha_1)$ into a task network $tn_2 = (T_2, \prec_2, \alpha_2)$ if $t \in T_1$ with $\alpha_1(t) = c$ and there is a task network $tn' = (T', \prec', \alpha')$ with $tn' \cong tn$ and $T_1 \cap T' = \emptyset$. The resulting task network tn_2 is defined as

$$\begin{aligned} tn_2 &= ((T_1 \setminus \{t\}) \cup T', \prec' \cup \prec_D, (\alpha_1 \setminus \{t \mapsto c\}) \cup \alpha') \\ \prec_D &= \{(t_1, t_2) \mid (t_1, t) \in \prec_1, t_2 \in T'\} \cup \\ &\quad \{(t_1, t_2) \mid (t, t_2) \in \prec_1, t_1 \in T'\} \cup \\ &\quad \{(t_1, t_2) \mid (t_1, t_2) \in \prec_1, t_1 \neq t \wedge t_2 \neq t\} \end{aligned}$$

To denote that the task network tn can be decomposed into the task network tn' by the application of one or more decompositions, we will write $tn \to^* tn'$.

The state-based part of the problem is defined over a set of propositional environment facts L. Primitive tasks hold preconditions and effects that change the current state. These are given by the functions in the triple $\delta = (prec, add, del)$, defining their preconditions, the positive and negative effects. All these functions are defined as $f : A \to 2^L$. The function $\tau : A \times 2^L \to \{true, false\}$ returns whether an action is applicable in a state with $\tau(a, s) \Leftrightarrow prec(a) \subseteq s$. When an action is applicable, the state resulting from its application is defined by the function $\gamma : A \times 2^L \to 2^L$ with $\gamma(a, s) = [s \setminus del(a)] \cup add(a)$. An action sequence $\langle a_0 a_1 \dots a_n \rangle$ is applicable to a state s_0 when each action a_i with $1 \leq i \leq n$ is applicable to the state $s_i = \gamma(a_{i-1}, s_{i-1})$.

An HTN planning domain is a tuple $D = (L, C, A, M, \delta)$. A planning problem $P = (D, s_0, tn_I)$ includes a domain D, the initial state $s_0 \in 2^L$ and an initial task network tn_I. A task network $tn_S = (T_S, \prec_S, \alpha_S)$ is a solution to a planning problem P if and only if the following conditions hold:

1. $\forall t \in T_S : \alpha_S(t) \in A$, i.e., all tasks are primitive.
2. There is a sequence $\langle t_1 t_2 \dots t_n \rangle$ of the task identifiers in T_S in line with $\prec_S$ and $\langle \alpha_S(t_1) \alpha_S(t_2) \dots \alpha_S(t_n) \rangle$ is applicable in s_0, i.e., the solution is executable.
3. $tn_I \to^* tn_S$, i.e., the solution is a refinement of the initial task network.

$Sol(P)$ denotes the set of all solutions to a problem P.

2.2 The Plan and Goal Recognition Problem

Based on the formal framework of HTN planning, we specify the PGR problem. Let $D = (L, C, A, M, \delta)$ be an HTN planning domain that defines the used model of behavior.

Having a hierarchical behavior model, goals of an agent are commonly defined as the *top-most tasks* in that hierarchy. Since an agent may pursue more than one of these tasks, we define goals in terms of a *task network*. This enables a flexible definition of goals, including multiple tasks that may or may not include the same task more than once. They may be ordered to each other, but if they are not, the approach allows for interleaving between the plans of several goals.

Definition 1 (PGR Problem) *A* PGR problem $(D, s_0, \bar{o}, \mathcal{G})$ *extends the* HTN *domain model by an initial state* $s_0 \in 2^L$*, the observations* $\bar{o}$*, and a set of possible* goals $\mathcal{G}$*. Let* $\bar{o} = \langle o_1, o_2, \dots, o_m \rangle$ *be the sequence of observed task names (the observations). We define* $\mathcal{G}$ *to be a (finite) set of task networks* $\mathcal{G} = \{G_1, G_2, \dots, G_r\}$ *where each element is defined as* $G_i = (\{id_0, id_1, \dots, id_n\}, \prec, \alpha)$ *with* $\alpha(id_j) \in C \cup A$ *and* $n \in \mathbb{N}_0$.

Since o needs to be placed at the beginning of the solutions, we will also call it the *prefix* of observations. $\mathcal{G}$ contains all possible combinations of top-most tasks. Technically, this means that any $G \in \mathcal{G}$ may form the initial task network.

Before we define the solution to such a problem, we have to discuss if we want the enforced prefix to be *uninterrupted*, or if it may include additional actions in between. To allow such additional actions may have different reasons:

- Ramírez and Geffner (2010) allowed for such interruption to deal with several goals with interleaving plans. However, since we support multiple goals in the *initial task network*, this is not necessary in the HTN setting.
- When the environment is partial observable, some actions may be missed. For such scenarios, it would be useful to allow actions in the prefix of our plans that have not been observed. We consider this option as important next step for future work. In the following, we assume that there are no such unobserved actions.

Definition 2 (Recognized Goals and Plans) *Given a PGR problem* $(D, s_0, \bar{o}, \mathcal{G})$*, some* goal $G \in \mathcal{G}$ explains *the observations* $\langle o_1, o_2, \dots, o_m \rangle$ *if and only if there is a task network* $(T, \prec, \alpha) \in Sol((D, s_0, G))$ *and an executable linearization* $\langle t_1, t_2, \dots, t_n \rangle$ *of the tasks in* T *with* $n \geq m$ *and* $o_i = \alpha(t_i)$ *for* $1 \leq i \leq m$*. We call* $\langle \alpha(t_1), \alpha(t_2), \dots, \alpha(t_n) \rangle$ *the recognized* plan.

3 Plan & Goal Recognition as HTN Planning

This section presents our approach that, like related work, is based on a problem transformation. However, due to the HTN setting, the overall approaches differ and enforcing a prefix is more complicated. Figure 1 illustrates the overall approach. The transformation is a two-stage process, a first step introduces a new task name t_I that can be decomposed in one of the original goal-networks $G_1 \dots G_r$. Every call of the planner starts with t_I in the initial task network. This transformation makes all solutions belonging to all goals $G_1 \dots G_r$ reachable. The system is free to choose one. Afterwards it needs to find a solution by using the decompositions defined in the original model (indicated by the cloud). A second transformation guarantees that any solution starts with the sequence of observations $\langle o_1 o_2 \dots o_n \rangle$.

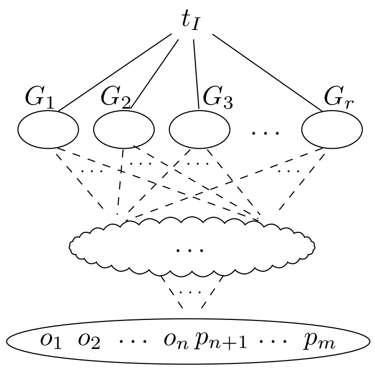


Figure 1: Schema of the overall approach. A new task t_I is the only task in the initial task network and must be decomposed into one of the original goal-networks G_i – the choice is made by the planner. Afterwards, G_i is further decomposed until a solution is found. A second transformation ensures that every solution starts with the observed prefix.

The planning problem that results from this transformation is an ordinary HTN planning problem that can be solved by any HTN planner. Solving the planning problem also solves the plan and the goal recognition problem: Any solution includes as first decomposition the goal G_i the planner has selected as well as the postfix of the plan that was generated – i.e., it provides a goal that explains the observation and it recognizes the plan.

The approach of Ramírez and Geffner chooses the goal in a designated step after planning. The transformation given in Sec. 3.1 enables us to integrate goal selection into the planning process and thus to solve the PGR with a single run of the planning system (instead of planning once per possible goal). The transformation that enforces the prefix is more difficult in HTN planning because, beside modifying and extending the set of actions, we need to adapt the decomposition hierarchy. Otherwise, the new actions are never reachable for the planner. The next sections introduce the two transformations in more detail.

3.1 Choosing a Goal

Based on a given PGR problem $(D, s_0, \bar{o}, \mathcal{G})$ with $D = (L, C, A, M, \delta)$, we define a planning problem

$$P = ((L, C \cup \{t_I\}, A, M', \delta), s_0, tn'_I)$$
$$M' = M \cup \{(t_I, tn) \mid tn \in \mathcal{G}\}$$

where $\{t_I\} \cap (C \cup A) = \emptyset$ is a new initial task and $tn'_I = \{\{id\}, \emptyset, \{(id \mapsto t_I)\}\}$. In the original model, $\mathcal{G}$ formed the set of possible initial task networks. Now exactly those networks can be decomposed from the newly introduced task. Obviously, the following lemma holds:

Lemma 1 *The set of solutions $Sol(P)$ contains* exactly *the solutions to all goals given in the original model* PGR.

Next, the problem is transformed so that the set of solutions includes solely those that start with the given prefix.

3.2 Enforcing a Prefix

We build on a transformation of the *actions* similar to that known from classical planning. Then we show how to integrate the new actions into the decomposition hierarchy to make them reachable from the initial task network.

Based on a given HTN planning problem $P = ((L, C, A, M, \delta), s_0, tn_I)$ with $\delta = (prec, add, del)$, we define a problem $P' = ((L', C', A', M', \delta'), s'_0, tn'_I)$ with $\delta' = (prec', add', del')$. Let $\bar{o} = \langle o_1, o_2, \dots, o_m \rangle$ be the sequence of observations.

To enforce the given plan prefix, we need to introduce new propositional symbols and duplicates of the actions with altered preconditions and effects. Let l_i with $0 \leq i \leq m$ and $l_i \notin L$ be proposition symbols that are used to place some o_i at its position in a generated plan, i.e. $L' = L \cup \{l_i \mid 0 \leq i \leq m\}$. For each task name o_i in the prefix, we introduce a new task name o'_i and define the preconditions and effects as

$$prec'(o'_i) \mapsto prec(o_i) \cup \{l_{i-1}\},$$
$$add'(o'_i) \mapsto add(o_i) \cup \{l_i\} \text{ and}$$
$$del'(o'_i) \mapsto del(o_i) \cup \{l_{i-1}\}.$$

Every action in the original problem needs to be executed after the prefix, i.e., $\forall a \in A$ holds that $prec'(a) \mapsto prec(a) \cup \{l_m\}$. The new set of actions is defined as $A' = A \cup \{o'_i \mid 1 \leq i \leq m\}$. To make the first action of the prefix applicable in the initial state, the symbol l_0 is added, i.e., $s'_0 = s_0 \cup \{l_0\}$.

We want every solution to include the entire prefix. Therefore we enforce l_m to be true at the end of each plan. Due to the lack of a goal description in the HTN formalism, we introduce a new primitive task name t_G with $prec'(t_G) \mapsto \{l_m\}$, $add'(t_G) \mapsto \emptyset$ and $del'(t_G) \mapsto \emptyset$. We place this task via the initial task network after all other tasks $tn'_I = (\{id_1, id_2\}, \{(id_1, id_2)\}, \{id_1 \mapsto t_I, id_2 \mapsto t_G\})$. Now, the HTN system is forced to search for plans that end with this action that holds our new goal as precondition.

So far we adapted the non-hierarchical part of the problem, but the newly introduced actions would never be reachable through decomposition, and due to the new precondition, the original actions would never be executable.

Regarding the newly introduced actions o'_i as equal to the actions they are duplicates of, it should be possible to place them at any position in the plan where o_i could have been. Therefore we introduce new abstract tasks that replace the primitive tasks in the original problem. New methods decompose these tasks either into (1) the original primitive task, or, (2) one of the new primitive tasks in the prefix.

$$C' = C \cup \{c'_a \mid a \in A\}, c'_a \notin C \cup A,$$
$$M^c = \{(c, (T, \prec, \alpha')) \mid (c, (T, \prec, \alpha)) \in M\}, \text{ where}$$
$$\forall id \in T \text{ with } \alpha(id) = n, \alpha'(id) = \begin{cases} n, & \text{if } n \in C \\ c'_n, & \text{else.} \end{cases}$$
$$M^a = \{(c'_a, (\{id\}, \emptyset, \{id \mapsto a\})) \mid \forall a \in A\},$$

Now we have to introduce a new method for every action in the sequence of observations $\bar{o} = \langle o_1, o_2, \ldots, o_m \rangle$. This method decomposes the respective new compound task c'_{o_i} into the observed action o_i:

$$M^o = \{(c'_{o_i}, (\{id\}, \emptyset, \{id \mapsto o'_i\})) \mid o_i \in \bar{o}\}$$

By defining the new set of methods $M' = M^c \cup M^a \cup M^o$, the new planning problem P' is fully specified.

Given that (1) the original primitive tasks and their corresponding duplicates are regarded equal and (2) the last artificial goal task t_G is deleted from the solution, the following property holds:

Lemma 2 *Given an* HTN *planning problem P and a prefix $\bar{o}$. Let P' be the transformed* HTN *problem that enforces the prefix. $Sol(P')$ contains* exactly *the solutions of P starting with $\bar{o}$.*

3.3 Overall Transformation

Given a PGR problem $(D, s_0, \bar{o}, \mathcal{G})$, we compile D, s_0, and $\mathcal{G}$ into a planning problem P by using the first transformation and enforce $\bar{o}$ by the second transformation, resulting into P'. Taking Lemma 1 and Lemma 2 together, we get:

Corollary 1 *Let* PGR *be a plan and goal recognition problem and P' an* HTN *planning problem resulting from applying the transformations. Then, $Sol(P')$ contains* exactly *the solutions to the original problem* PGR.

Instead of constructing a specialized plan and goal recognition system to solve the task, we can create the corresponding HTN problem and pass it on to any HTN planning system. The next section gives empirical results on that approach.

4 Empirical Evaluation

We implemented a lifted version of the transformation and used a variant of the PANDA planning system (Bercher, Keen, and Biundo 2014) to solve the resulting problems. It is not easy to find evaluation domains for PGR. Beside the actual model for the recognition, one needs to have pairs of goals and plans that belong to each other. Moreover, interesting domains should incorporate a set of qualitatively different goals. We have chosen the Monroe domain[1], a disaster management domain built by Blaylock and Allen (2005) to generate a corpus of 5000 plan/goal combinations. They generated the plans by using a modified SHOP2 planning system that was adapted to randomly generate different possible plans for a given goal. It includes 10 different (lifted) goal-networks (each containing a single goal-task), 46 (lifted) methods and 30 (lifted) actions as well as a huge set of constants that result in nearly 2000 distinct groundings of the goals in the generated corpus. For details, we refer to their paper given above. From a computational complexity view, the domain is neither totally ordered, nor (grounded/lifted) acyclic, nor tail-recursive, i.e., properties that would make it decidable (Erol, Hendler, and Nau 1996; Alford, Bercher, and Aha 2015), are not fulfilled. We use an HTN domain that is widely based on their SHOP2 domain.

[1]https://www.cs.rochester.edu/research/speech/monroe-plan/

We use the first 100 experiment runs of the corpus. For each run, we generated one planning problem with enforced prefix of length 0, 1, 2, and so on, until the whole original plan has been enforced as prefix. The plans of the used instances have a length of 4 to 29 actions with a mean value of 10.7. We ended up with a total of 1174 distinct planning problems and ran each with 10 random seeds, i.e., the given results are based on 11740 runs of the planning system (on Xeon E5-2660 v3 CPUs with 2.60 GHz base frequency, 32 GB memory limit and 5 minutes time limit).

The system solved 94.4 % of the planning problems within this limit. For the solved instances, the planning system needed a mean runtime of 57 seconds with a standard derivation of 17.8. Due to the vast set of constants in the domain, most of the time was spent on grounding the problem.

4.1 Goal Recognition

Figure 2a shows the performance of goal recognition over all solved instances and all goals. On the x-axis it shows the length of the observed prefix, on the y-axis the percentage of instances with correctly classified goal. Due to the different plan lengths, the length of the prefix is given as *percentage* of the overall plan. Each point represents the ratio between instances with correctly recognized goal and instances with wrongly recognized goal in an interval of 12.5% of the prefix length. Consider, e.g., the right-most triangle in the plot at position (93.75, 86.4). It means that, from instances with an enforced prefix of a length between 87.5% and 100% of the ground-truth plan, 86.4% are solved correctly.

The top-most curve shows the recognition rate on the goal-task *without parameters* (i.e. 10 possible classifications). Here the goal is recognized correctly after a short prefix. The curve at the bottom shows the recognition rate *with parameters*, i.e., the result is correct when the task and all parameters have been recognized correctly. This results in thousands of possible classifications (nearly 2000 of them included in the corpus). We will come back to the third curve later.

Studying the first plots, we were wondering why the system could not detect the correct parameters for the goal in so much cases. So we had a look at the performance on distinct goals. Naturally, there are goals that result in sequences of actions that are more unique than others, revealing the goal. Other goals may result in plans that do not really distinct one goal from another. So one might assume that some goals are

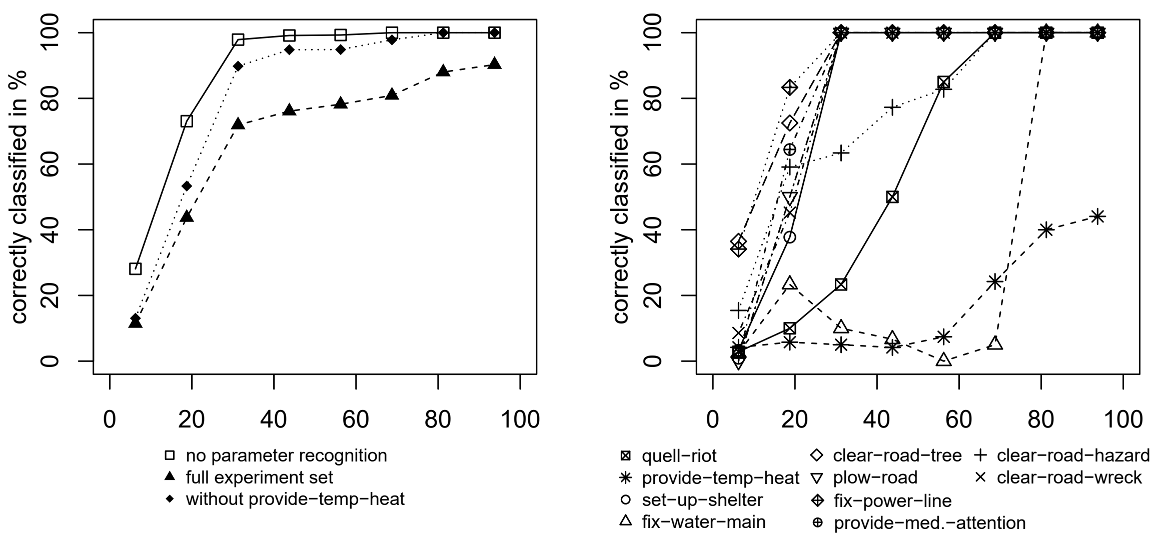


Figure 2: The percentage of correctly classified goals. (a) over all goals. (b) split by (ground-truth) goal.

recognized after a short prefix, others might need a longer prefix to be recognized.

Figure 2b shows the recognition results split by the 10 different types of goal. Some of the goals are recognized quite well after observing a short prefix, others only when the half or even three quarters of the overall plan have been observed. Quite notable is the poor recognition performance on the goal *provide-temp-heat*. We identified the following method in the domain used for generation that does cause it:

```
(:method m-provide-temp-heat-1
  :parameters (?person - person
               ?ploc - point)
  :task (provide-temp-heat ?person)
  :precondition (and
    (atloc ?person ?ploc))
  :ordered-subtasks (and
    (generate-temp-electricity ?ploc)
    (turn-on-heat ?ploc)))
```

It decomposes the task *(provide-temp-heat ?person)* into two subtasks and includes a method precondition that binds the parameter *?ploc* to the location of the person to treat. The subtasks ensure that this location is provided with electricity and is heated, but the person itself does not appear in the plan. Given that more than one person may be at this location, it is impossible for a PGR system to infer the right one. Surely, such structures should be avoided when designing a model for plan and goal recognition (so it might be totally fine for planning).

Let us have another look at Fig. 2a. The curve in the middle gives the performance over all goals except for *provide-temp-heat*. Now the overall performance of recognizing the goal with all parameters nearly reaches the performance of recognizing only the type of the goal without parameters.

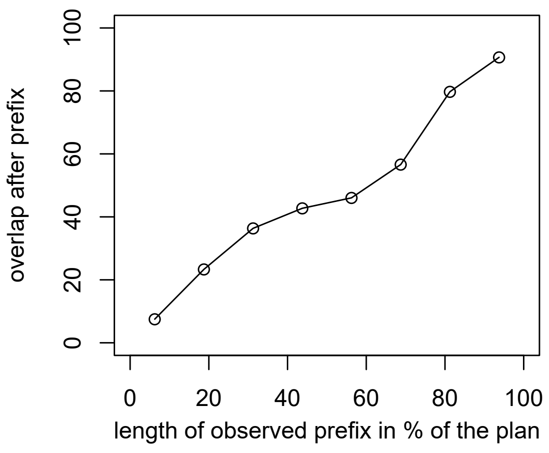
4.2 Plan Recognition

Plan recognition provides further information about the observed agent's behavior and might support the activity recognition (that recognizes the observed actions) by giving a prior on which actions might be observed next.

The ground-truth plan and the recognized plan have necessarily the enforced prefix of observations in common. Figure 3a gives the percentage of the actions in the *postfixes* that are identical (over all runs, regardless on whether the goal was recognized correctly or not).

In general, the plan generated by the planning system might be longer than the ground-truth plan. Therefore, the given percentage is calculated in the following way: Given the number of common actions after the prefix is c, the ground-truth plan has length g, the recognized plan length r and the prefix length p, the given percentage is $100/max(g-p, r-p) * c$.

Please be aware that inferring a total ordered sequence of task that can be observed next might be difficult due to the partial ordering in task networks. However, we want to predict the directly forthcoming actions to support the activity recognition. Figure 3b shows the percentage of runs where the next (up to) three actions have been inferred correctly, given a certain prefix. The curves stand for outlook lengths of one ($\circ$), two ($\triangle$), and three ($+$). For example, the circle located at position (6.25, 21.26) means that from all runs (goal correct or not) where 0 up to 12.5 percent of the plan's prefix has been enforced, the action after the prefix has been classified correctly in 21.26% of the runs. We want to note one technical detail of this last diagram. The planner is in principle free to generate plans that are longer than the ground-truth plan (since it has no information that the full prefix was enforced). Therefore we do not want to punish the fact that the recognized plan has the same size as the ground-truth plan and count the identified end of the plan also as correct classification.



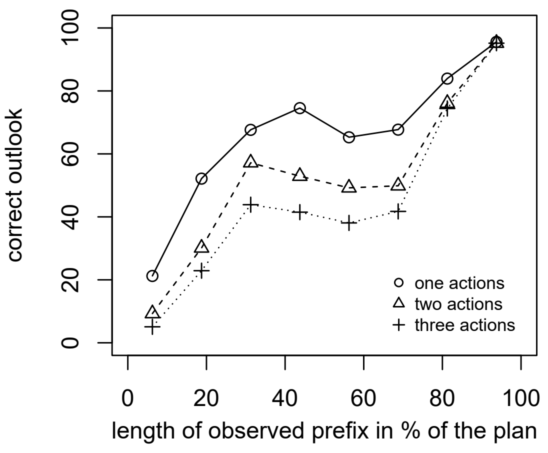


Figure 3: (a) Percentage of correctly recognized actions *after* the enforced prefix over *all* runs. (b) Percentage of runs where the first one, two, or three action after prefix are correct.

Resuming the plan recognition results, it can be seen that, though the evaluated domain enables a large set of possibilities on how to achieve a goal and the way the corpus was generated ensures that many of them are included, the system finds plans with a large percentage of actions that are included both in the generated as well as in the ground-truth plan, so that the overall performance should help the activity recognition and should also allow to gain knowledge about the agent's future course of action.

5 Discussion

An interesting question in PGR is which assumptions are made about the observed agent and the environment. Like the work of Ramírez and Geffner (2009; 2010), we rely on full information about the initial state and deterministic actions. We also assume fully observable actions, but it is straightforward to extend our approach to deal with missed observations.

Concerning the agent, it needs to be specified which course of action is regarded a *rational* way of reaching a goal. In other words: which behavior is regarded *goal directed*. Ramírez and Geffner (2009) assume the agent to execute an *optimal* plan, otherwise the goal that is approached in a non-optimal way is rejected from the set of possible goals. Ramírez and Geffner (2010) do a less restrictive assumption. They do not require optimal behavior. Instead, the approach is based on the *cost difference* between the plans with/without enforced prefix. So the agent needs to approach the goal, but not necessarily in an optimal way. In HTN models, the courses of action are more restricted, since it is not possible to insert actions apart from decomposition. In fact, the hierarchy is often regarded to specify combinations of actions that are *helpful strategies* to reach a goal. So the assumptions about goal-directed behavior are given in the domain. Nevertheless could our approach benefit from additional metrics, especially to rank goals against each other. Such heuristics may be defined on the length of the resulting primitive plan, but also on the applied methods.

Currently, our approach excludes unreachable goals and picks one of the remaining, this is more closely related to Ramírez and Geffner (2009) than to (2010). Here, the observations are *"... replaced by extra goals that must be achieved at no extra cost."* (Ramírez and Geffner 2009, p.1780) and optimal plans are needed. They call it a "filter" resulting in the set of goals in line with the observations. In principle, such filtering is also possible with our prefix-transformation by the cost of planning for each goal – we wanted to avoid this effort. However, when our full transformation is applied, the used HTN planning system sorts out many unreachable parts of the search-space during the process of grounding. Though based on *relaxed* reachability (so there might remain goals that are unreachable), this may be an interesting information.

Since the HTN formalism has (like STRIPS) been developed to represent agent behavior, we think that it is a good choice to model behavior in plan and goal recognition. The combination of hierarchy and partial ordering between tasks makes the formalism very expressive: it has been shown that the models can represent (non-context-free) context-sensitive languages (Höller et al. 2014), whereas STRIPS can only express regular languages (Höller et al. 2016). We think that especially the partial ordering is useful in PGR: it enables an observed agent to pursue more than one goal and to interleave the plans. It might also be used when multiple agents are observed to recognize multi-agent plans.

There are two interesting variations of the formalism that might be considered for future work: one allows shared tasks between the plans of several goals (see Alford et al. 2016). However, such behavior might also be modeled via no-op operators; and it has not been shown yet how this changes the expressivity. A second variant allows the planning system to insert tasks apart from the hierarchy (Geier and Bercher 2011). This enables noise in the observation (e.g. when the observed agent executes actions that are between the goal-directed actions or the activity recognition has been wrong). However, using this variant, the planning system is no longer forced to integrate observations into the plan belonging to the recognized goal, so some metric is needed to enable the insertions that are needed, but punish plans with too much insertions. Otherwise it is not possible to choose, e.g., between some (maybe in general unlikely) solution that starts with exactly the observed actions and a (in general more likely) plan that does not. Task insertion can also be compiled into ordinary HTN domains – it has been shown that the resulting models are far less expressive than pure hierarchical models (Geier and Bercher 2011; Höller et al. 2014).

Throughout the paper, we assumed that a set of possible goals are given in form of a set of task networks. This resulted in the limitation that the number of this goals has to be finite (otherwise we end up with an infinite set of methods). If necessary, the given construction to choose the goal of the agent can be replaced by one that enables a recursive insertion of goal-tasks (e.g. by starting with a single task as goal and having a method that replaces this single task by two new tasks). That way, one can introduce an arbitrary number of goals pursued by the agent (though we do not think that having a finite set of goals is a limitation in practice).

6 Conclusion

We showed how the approach *PGR as Planning* can used in the context of HTN planning. *PGR as Planning* enables the field of PGR to benefit from present and future research in planning. By using HTN instead of STRIPS models, our approach enables the use of far more expressive types of models to define the recognized behavior (more precisely, context-sensitive instead of regular languages), thereby combining the widely-used grammar-like models with state-transition as given in STRIPS. We improved the approach in a way that only one run of the planning system is necessary, regardless of how many possible goals there are. The choice on which goal the observed agent pursues is made by the planning system – this enables the use of its pruning and search techniques when choosing the goal. The empirical evaluation showed that the approach works well on an evaluation domain containing a huge set of possible goals; that current HTN planning systems are able to solve the transformed problem quickly and that the approach results in good recognition rates.

Acknowledgments

This work was done within the Transregional Collaborative Research Centre SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems" funded by the German Research Foundation (DFG).

References

Alford, R.; Bercher, P.; and Aha, D. W. 2015. Tight bounds for HTN planning. In *Proc. of the 25th International Conference on Automated Planning and Scheduling (ICAPS)*, 7–15. AAAI Press.

Alford, R.; Shivashankar, V.; Roberts, M.; Frank, J.; and Aha, D. W. 2016. Hierarchical planning: Relating task and goal decomposition with task sharing. In *Proc. of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 3022–3029. IJCAI/AAAI Press.

Behnke, G.; Höller, D.; and Biundo, S. 2015. On the complexity of HTN plan verification and its implications for plan recognition. In *Proc. of the 25h International Conference on Automated Planning and Scheduling (ICAPS)*, 25–33. AAAI Press.

Bercher, P.; Keen, S.; and Biundo, S. 2014. Hybrid planning heuristics based on task decomposition graphs. In *Proc. of the 7th Annual Symposium on Combinatorial Search (SOCS)*, 35–43. AAAI Press.

Blaylock, N., and Allen, J. F. 2005. Generating artificial corpora for plan recognition. In *Proc. of the 10th International Conference on User Modeling (UM)*, 179–188. Springer.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1996. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence* 18(1):69–93.

Geib, C. W., and Goldman, R. P. 2011. Recognizing plans with loops represented in a lexicalized grammar. In *Proc. of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, 958–963. AAAI Press.

Geib, C. W. 2004. Assessing the complexity of plan recognition. In *Proc. of the 19th National Conference on Artificial Intelligence, 16th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, 507–512. AAAI Press/The MIT Press.

Geier, T., and Bercher, P. 2011. On the decidability of HTN planning with task insertion. In *Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, 1955–1961. AAAI Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2014. Language classification of hierarchical planning problems. In *Proc. of the 21st European Conference on Artificial Intelligence (ECAI)*, 447–452. IOS Press.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2016. Assessing the expressivity of planning formalisms through the comparison to formal languages. In *Proc. of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*, 158–165. AAAI Press.

Ramírez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 1778–1783. IJCAI/AAAI Press.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 1121–1126. AAAI Press.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan recognition as planning revisited. In *Proc. of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 3258–3264. IJCAI/AAAI Press.

Sukthankar, G.; Goldman, R. P.; Geib, C.; Pynadath, D. V.; and Bui, H. H. 2014. *Plan, Activity, and Intent Recognition*. Elsevier. chapter An Introduction to Plan, Activity, and Intent Recognition.