

# Incremental Case-Based Plan Recognition Using State Indices

Boris Kerkez and Michael T. Cox

Department of Computer Science and Engineering  
College of Engineering & CS  
Wright State University  
Dayton, OH 45435-0001

{bkerkez;mcox}@cs.wright.edu

**Abstract.** We describe a case-based approach to the keyhole plan-recognition task where the observed agent is a state-space planner whose world states can be monitored. Case-based approach provides means for automatically constructing the plan library from observations, minimizing the number of extraneous plans in the library. We show that the knowledge about the states of the observed agent's world can be effectively used to recognize agent's plans and goals, given no direct knowledge about the planner's internal decision cycle. Cases (plans) containing state knowledge enable the recognizer to cope with novel situations for which no plans exist in the plan library, and to further assist in effective discrimination among competing plan hypothesis.

## 1. Introduction

In plan recognition systems, the main performance task is recognition of an observed agent's plans and goals, based on agent's planning behavior. Recognition can be *intended*, as in natural language dialogue [1, 6], where the observed agent is aware of the recognizer, and tries to assist (cooperative) or obstruct (adversarial) the plan recognition task. During *keyhole* plan recognition the observed agent does not participate in the recognition process; the recognizer determines the goals of the planner based solely on available observations of planner's behavior. In order to infer the plans and goals of the observed agent, the recognizer typically compares the observations of planner's behavior with possible plans contained in its *plan library*, and tries to find the plan(s) from the library that would account for observed behavior.

In a large number of plan recognition systems [4, 11], the plan library is specified for the recognizer *a priori*, by some external agent who is often the system designer. This limits the plan recognition process in that only the plans known in advance can be recognized, and novel plans will not be accounted for. Furthermore, it is often required that the plan library be complete [4, 17], i.e., the plan library contains all of the possible plans an observed agent may pursue. This approach is suitable in situations where possible plans can be enumerated in advance. However, enumerating

all of the plans in large domains can be a difficult knowledge acquisition task, as the number of possible plans may be very large. Furthermore, this enumeration may include those plans that the planner never uses. [12] shows how the presence of extraneous plans in the plan library can impact the efficiency of the recognizer. They also introduce techniques to automate the process of constructing the plan library by synthesizing possible actor's plans and goals through plan and goal biases. The automatic plan library construction methods suffer from the same problem as the hand-coded plan libraries, since the plans that are automatically generated may be extraneous (even though the biases will minimize their occurrence).

We present a case-based approach to keyhole plan recognition applicable to state-space planners where world states of the planner can be monitored and where states are represented using the first-order predicate calculus. We focus not only on the knowledge about the actions the planner performs, but also on the states of the world the planner finds itself in during the planning. We show that the knowledge about the world states increases the efficiency of the recognizer in planning domains with wealth of the state information. Cases rich in state knowledge enable more informed predictions for recognition systems in which the planner's internal decision cycle (i.e. reasons for taking actions) is not exposed. We propose a scheme intrinsic in the case-based approach, in which a plan recognition system is able to learn about novel plans and incorporate them in its plan library for future use, thus incrementally improving its recognition process. By employing a case-based approach to plan recognition along with the knowledge about planner's world states, our system may be able to predict the agent's behavior even when the actions and states observed are not consistent with any plans in the plan library. This is because our system is able to partially match past cases, which guide the recognition process when exact matches are not available. This approach greatly improves the robustness and flexibility of a plan recognition system.

The next section describes the motivation behind our work. Section 3 deals with the benefits of retaining state knowledge, which is used in section 4 where concepts of indexing and case similarity are explored. Section 5 presents implementation and recognition examples. Finally, we conclude with the summary of presented topics in section 6.

## **2. Motivation and Related Work**

In our opinion, the case-based approach comes as a natural reasoning paradigm for the plan recognition task. The plan library contains instances of plans an observed agent may pursue. Planning episodes in the plan library can be viewed as cases, and the recognition process can utilize these past plans (cases) to generate predictions. The main focus in the context of plan recognition is on retrieval of cases that account for the observed behavior, and not on the adaptation of the retrieved cases to solve planning problems. While most traditional plan recognition systems [11] are able to recognize only those plans whose steps are exact matches to the plans in the plan library, a pure case-based approach to plan recognition allows the recognizer to partially match cases (plans) from the library with its observations. A case-based recognizer can therefore recognize plans that are not exact matches to the plans from the

library, but instead are similar to the description of a current situation, based on the observed planning steps.

In our system, cases (plans) contain not only knowledge about actions, but also the knowledge about the states of the world seen during the planning. Such state knowledge can be of great utility for case-based plan recognition, because, cases retrieved on a basis of matched planning actions can further be narrowed down by their applicability, given the current state of the world. Bares and his colleagues [5] explore the concept of case-based plan recognition, but, their cases do not represent explored world states like our system does.

A case-based plan recognizer can efficiently overcome difficulties of requiring a complete plan library specified for the recognizer *a priori*. The recognizer can start the recognition process with an incomplete (or empty) plan library and accumulate the plans from observations of the planner's behavior. The observed plans themselves constitute contextualized pieces of knowledge useful for future predictions from observations. The main disadvantage of this approach is that some of the plans observed for the first time may not be recognized by the system initially, because relevant plans may be absent from the plan library. However, the observed plans are then learned and used by the recognizer in subsequent plan observations. The ability to incorporate novel plans into a plan library makes a case-based recognizer applicable to a wide variety of different applications, especially to those where all of the plans may not be known in advance and therefore cannot be enumerated (e.g. recognizing military maneuvers of enemy ground troops).

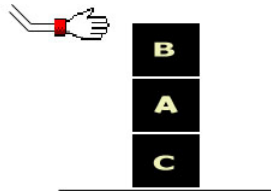
The ability of a plan recognition system to deal with an incomplete plan library is important, because, constructing a complete plan library can be a difficult knowledge acquisition task. Enumeration of all of the plans in the library can impact the efficiency of the recognizer, because, such an enumeration may include extraneous plans, which the planner will never execute and the recognizer will need not to predict. The methods for automatic plan library construction may suffer from the same problem. Our incremental construction of the plan library ensures that only the plans that were actually observed during the planning process are used in the recognition process, and therefore minimizes the number of extraneous plans in the library. If we assume that the planner is a goal-oriented expert (all performed planning steps are necessary to reach the goal), we are guaranteed that the plan library will contain no extraneous plans.

### 3. State Knowledge

The focus of our research is to incorporate the knowledge about the states of the world in which the planner finds itself during the planning and the plan recognition processes. Traditionally a plan is simply a sequence of actions that transform the initial state into the goal state. However our system represents a case (i.e., a plan) as a sequence of action-state pairs. Such pairs encode both the action and the new state that results from an action. Note that using this representation, the initial pair in a plan is always the *null* action that "results" in the initial state, and, the final pair is the action that results in the goal state.

The knowledge about the world states allows for more informed recognition of a planner's goals than if the system relied on past actions alone. Competing hypothesis about possible pursued plans and goals based on past actions can further be discriminated on the basis of the current state of the world, as some consistent plans may not be applicable in the current world state. As an example, consider the logistics planning domain, where packages are supposed to be transported to their destinations, which can be either a post office or an airport on an island. Airplanes fly between islands (e.g., from an airport on one island to another), and, trucks transport packages within islands (e.g., between an airport and a post office). In this domain, for example, plans anticipating packages to be loaded into an airplane as the next step may be eliminated from the set of possible hypothesis, given a world state in which no packages are present on an island at which the airplane is. Other researchers have explored the role of the state knowledge to a limited degree. Albrecht and his colleagues show different Bayesian net recognition models, one of which recognizes plans based on the state knowledge. They postulate that such a model would be applicable in domains with limited number of actions [2]<sup>1</sup>. Their work is different from ours in that the states they encounter are simplified<sup>2</sup> and the number of different states is relatively small (about 4000).

In order to be able to observe the state of the world during the planning, we focus on state-space planners in which actions change the state of the world [8]. States are represented as sets of instantiated first-order logical predicates (or *ground literals*). For example in the blocksworld domain,  $\{(clear\ B), (on\ B\ A), (on\ A\ C), (on-table\ C), (arm-empty)\}$  represents the state depicted in figure 1.



**Fig. 1.** Example of a blocksworld state  $\{(clear\ B), (on\ B\ A), (on\ A\ C), (on-table\ C), (arm-empty)\}$ .

The most common notion of a plan is an ordered sequence of actions that transforms an initial state of the world to some distinguished goal state. Each action is represented by an operator that has preconditions and effects. Preconditions establish applicability criteria, and, effects change the state of the world. Furthermore, each action starts in a certain state of the world and results in a change from the current state to another one. We extend the notion of a plan to a sequence of state changes, where the state changes are caused by the actions the planner performs. Although

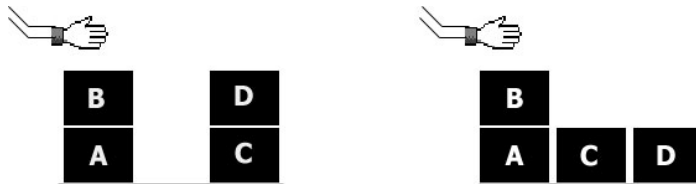
<sup>1</sup> We are referring to *locationModel*.

<sup>2</sup> States in their research are locations of a player in the context of online dungeon game. These states lack the detail of information provided by combinations of literals corresponding to sensory information.

world state changes in the real-world systems can be caused by exogenous events and external agents [e.g., 3, 16], we do not deal with this issue here.

Many planning systems (case-based or otherwise) [10, 13, 14, 15] also keep track of the world states; however, they commonly focus on a limited set of only two states explored by the planner. That is, planners concentrate on the initial state of the world and a specification of the goal state. Case-based planners are usually given a *partial* representation of the goal state in which the truth values of only some of all of the possible instantiated state predicates (literals) are specified. For example, a goal from the blockworld domain  $\{(on\ blockB\ blockA)\}$  lacks specifications for other state literals, such as positions of other blocks in the problem in hand. Such a *full* specification of a goal state allows for versatility and goal-directed planning, and permits more than one world state to achieve a goal (figure 2).

In our research, we assume that the plan recognition system is given no initial knowledge about partial goals of the planner. Because we focus on keyhole plan recognition, we are limited by available observations of the planner's activity during problem solving. Since state-space planners search the state-space during problem solving, we modified the PRODIGY planning cycle [7] to monitor the world states after the planner performs an action<sup>3</sup> to perform keyhole-like observation of the planner's behavior. Our cases contain all of the world states visited during the planning, along with all of the actions performed by the planner. The goal states in our system are described fully. Therefore, a keyhole-like observation of a goal state treats the goal as just another observed state.



**Fig. 2.** Both pictures depicted represent a state of the blockworld domain in which the partial goal  $\{(on\ B\ A)\}$  is satisfied. The complete state specification for the state on the left is  $\{(on\ B\ A), (on\ D\ C), (clear\ B), (clear\ D), (on-table\ A), (on-table\ C), (arm-empty)\}$ , while the state on the right is uniquely determined by  $\{(on\ B\ A), (clear\ B), (clear\ C), (clear\ D), (on-table\ A), (on-table\ C), (on-table\ D), (arm-empty)\}$  combination of literals. Eleven other world states, not depicted here, also satisfy goal  $\{(on\ B\ A)\}$ .

In the next section, we describe a scheme that is able to transform the observed states into their abstracted representation, retaining the knowledge about the structure of a type-generalized world state. The abstracted representation of world states allows us to focus the plan recognition process on the relevant features of the states, within much smaller abstract state-space.

A full representation of a goal state that accounts for state predicates other than ones explicitly given as traditional (partial) planning goals has another advantage

<sup>3</sup> We keep track of PRODIGY's *applied operators* during problem solving episodes; we focus only on those applied operators along the solution path. The monitoring is not implemented as those monitors described in [16].

in both case-based planning and plan recognition. A case-based planner may not be able to successfully retrieve a solution to the current problem when only partial goal state predicates are specified. For example, if we assume that the planner encounters a goal specification  $\{\text{NOT}(\text{clear blockA})\}$  for the first time, matching this goal state with states depicted in figure 2 would fail, even if problems in figure 2 were already solved. However, both states in figure 2 implicitly contain state predicates, which define a goal state for the newly seen problem; storing all of the state predicates for goal states would yield successful matches.

Although knowledge about the world states is beneficial for the plan recognition process in state-space planners, the number of possible states in the complete state-space may be quite large. The exact number depends on the domain theory as well as on the instances of objects a particular problem definition may entail. To gain understanding of how large the state-space may be, let us introduce

$$P^i = \{p \mid p \in D \wedge p \text{ is a predicate} \wedge |\arg s(p)| = i\}$$

a set of all predicates from the domain theory  $D$  that have arity  $i$ . Each predicate  $p$  has arguments of certain types from the type hierarchy of the domain theory, and, predicates instantiate into ground literals of appropriate types at planning time. For a given problem, let  $n$  be the number of different instances, and to account for the worst-case scenario, assume that all  $n$  instances are of the same type. We operate under a closed-world assumption, where if a predicate is not known to be *true*, then it is explicitly assumed that the truth-value of a predicate is *false*.

The representation of state-space that accounts for the complete states may be expensive. Given

$$P^0, P^1, \dots, P^q, \text{ where } q = \max |\arg s(p_i)|, \forall p_i \in D$$

the size of a feature bit vector representing a state without any loss of state information in the state-space is

$$|\vec{F}| = P^0 + nP^1 + n^2P^2 + \dots + n^qP^q = \sum_{i=1}^q n^i P^i$$

and the size of the state-space

$$|SS| = 2^{|\vec{F}|}.$$

In the blocksworld domain, given only 10 different blocks to work with, the feature vector size is 19, and the state-space size is 524,288. Therefore the approaches based on pruning a fully constructed state-space [e.g. 9] are computationally extremely expensive, because the state-space size prohibits efficient pruning. Such approaches are promising only if they are able to prune large regions of the state-space at one time.

However, except for the simplest domains with small number of possible states, planners tend not to explore all of the state-space during the planning. In fact, the number of states a planner encounters is domain dependent and often constitutes only a fraction of the number of all possible states it may explore, as our experimental

results show. Moreover, some of the states in the state-space are simply impossible to achieve. For example, having two blocks stacked on top of each other prohibits both of them from being *clear*. Likewise, a block cannot be on top of itself. Such anomalous states can never be reached. Finally, some of the states may not be useful to the planner, although they may be possible to achieve. If several solutions to some task exist (one of which is the best) then the planner may always opt to prefer the best solution and ignore the traversal of the state-space along sub-optimal paths, thus never encountering any of the states traversed along the sub-optimal solution paths.

Our approach implements incremental construction of the state-space, rather than the pruning of the complete state-space. To ensure that only the states that are useful for the planner are present, we start with empty state-space, monitor the states in which the planner finds itself, and incrementally expand our state-space to include nodes representing only the observed states. The space savings in our approach are significant, because the recognizer ignores the states that are not useful to the planner.

#### 4. Indexing and Similarity

Two of the most important components of any case-based system are its indexing and retrieval mechanisms. The importance is amplified in the context of the plan recognition task, where a case-based recognizer typically does not try to adapt the cases for problem solving, but rather tries to recognize plans that are consistent with observed planning behavior. As described in Section 3, plans are equivalent to cases in our system. Each case describes the temporal ordering of actions in the plan, as well as the world states before and after each action is performed, which represents all of the information we are able to observe during the keyhole state-space planning. Although it would be very interesting to incorporate knowledge about the planning failures, our initial work focuses on the situation where observed plans achieve their goals, and, the recognizer deals only with the actions and world states along the solution path ignoring the rest of the planner's unsuccessful search. Such an approach mimics the behavior of an expert planning agent. It also enables the system to minimize the number of world states of which it needs to keep track, although knowledge of situations in which planning failures occurred would be useful in recognizing such subsequent failures (e.g. Hammond's failure anticipation). Our future research efforts will explore this aspect of failure-driven learning.

The indexing mechanism in our system is based on matching the current state of the world with the world states observed previously during planning. One can think of the world states (and actions taken) as the recognizer's sensory inputs, which describe a specific situation in which the planner finds itself. When the planner encounters a similar situation in the future, the recognizer can draw predictions given the knowledge about planner's behavior in known similar situations. As the number of states explored by the planner can be quite large, the naïve indexing-scheme in which a state is compared with every other state, can prove to be very inefficient. Our indexing scheme employs a simple abstraction, at level of which states can be more efficiently compared and retrieved.

The plan recognition algorithm observes the planner's current state of the world as well as the current action that produced the world state. Based on these observations and a partial plan observed so far, the recognizer retrieves the plans (cases) that account for the observations, and predicts possible goals and plans that the planner may be pursuing. The retrieved cases are ranked by their relevance. Once the goal state is reached, we collect information about system's predictions and evaluate its usefulness.

The case base consists of a few hash tables for indexing purposes and a graph-based representation of the abstracted state-space that is constructed from the observations of planner's search. To increase indexing efficiency, we employ a simple abstraction scheme in which states of the world are abstracted to their type-generalized state predicates [14]<sup>4</sup>. The abstract state representation is a non-negative integer vector in which each dimension represents a number of instances of a single type-generalized state predicate. For instance, the dimensions of abstract state feature vectors in the blockworld planning domain are depicted in fig. 3. Abstract states constitute vertices of the abstracted state-space graph, in which an edge from one abstract state to another indicates the observation of an action that changed the former world state into the latter one. Each case can be abstracted to represent a path in the abstracted state-space graph.

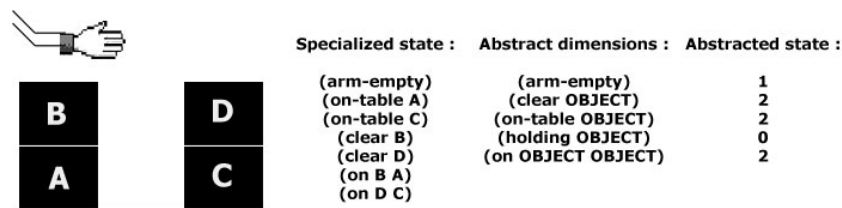


Fig. 3. An example of a blockworld planning domain state of the world and its representations. State predicates, whose truth values are false, are not shown in the *specialized state*.

A case is indexed by its abstracted state representation. Each abstract state points to a set of specialized world states (i.e., ground literals) indexed by it. These in turn point to the cases that contain them. Because we utilize hash tables for indexing cases and world states by abstract state representations, this scheme allows us to retrieve the initial set of matching cases very efficiently. When the recognizer tries to retrieve cases that match with observed plan steps, it first uses the abstract representation of the current state of the world to retrieve those cases that contain the current abstract state. If the current abstract state has never been observed before, the recognizer tries to find *similar* abstract states and returns the cases containing these states. If the recognizer cannot retrieve any such cases (i.e. no situations similar to the current one were observed in the past), it is unable to make predictions at this time and opts to wait for observations of subsequent planning steps. The use of abstract world

<sup>4</sup> State literal from the logistics planning domain, such as (at-obj A-300.1 JFK\_NY.1) would be type-generalized to (at-obj AIRPLANE AIRPORT), because the type of instances A-300.1 and JFK\_NY.1 are AIRPLANE and AIRPORT respectively.

states is an efficient indexing scheme at the top level, as the number of abstract states is much smaller than the number of specialized world states, and the distribution of specialized states into bins indexed by their abstracted representation provides means to eliminate a large number of possible hypothesis and focus the recognition process on the relevant cases.

More specifically, if no exact matches are found for the current abstract state, the recognizer attempts to find nodes in the state-space graph that are similar to the current abstract state as follows. Since abstract states are represented as feature vectors of positive integers, our system uses a modified nearest neighbor similarity metric, where the neighbor states considered are those which differ from the current abstract state by some distance in each dimension<sup>5</sup>. Similar abstract states are then ranked by their distance, and the abstract state at the shortest distance is used for retrieval.

After cases that matched at the abstract level are retrieved, the abstract states from the retrieved cases that matched with the current abstracted world state are compared at the specialized level (i.e. at the *ground literal* level). The similarity metric we currently use to match states is the same metric used in [14]<sup>6</sup>. Note that the indexing scheme at the level of abstracted world states ensures that all of the cases potentially similar at the specialized level will be considered. We are currently investigating the matching of cases at the specialized level by using a graph representation, where the specialized world states are encoded as directed graphs, and case matching is equivalent to the graph-isomorphism problem. This will increase the efficiency of the recognizer, because the complexity of the current approach is  $O(n!)$ . In contrast, there exist polynomial-time algorithms for graph-isomorphism for certain classes of graphs. Furthermore, since graph-isomorphism is an equivalence relation, the specialized world states corresponding to a single abstracted world state can further be indexed by their equivalence classes, thus reducing the amount of search during the retrieval.

If the set of possible plan hypothesis consists of a single case at this point, the case is used to guide the recognition process and form predictions. If more than one case is still a match, the recognizer uses the knowledge about the actions and past states in the matched cases to prefer the cases whose actions and states are more consistent with the plan observed so far. At present time, the matching scheme employs a graph matching algorithm, where cases can be abstracted into their abstract paths and matched among themselves.

The abstraction scheme provides a means of rapid access to the specialized world states indexed by their abstract representation. It is a one-to-many relation, because an abstract state may index several specialized states. The result is a smaller set of first level indexes, which ensures similarity at the level of specialized states. The exact space savings of such an abstraction scheme depend on the domain theory. In general, the number of possible abstract states for a given domain theory is

$$\prod_{i=1}^d m_i$$

---

<sup>5</sup> To increase matching efficiency, only the states that were actually observed within a ball with given radius are considered.

<sup>6</sup> The threshold we used in our similarity matching scheme was 2.

where  $d$  is the number of abstract dimensions, and  $m_i$  is the maximum possible value in abstract dimension  $i$ . Given  $n$  object instances, abstract dimensions corresponding to  $i$ -ary predicates may possibly have values from 0 to  $n^i$ . For example, in the case of the blocksworld domain, the abstracted state feature vector size is five (i.e., the number of abstract dimensions), and number of possible abstract states for ten object instances is 48,884. This represents a considerable savings when compared to the number of all possible specialized states (see section 3). As stated before, not all of these states are possible; in the blocksworld domain, there cannot be more than  $n-1$  instances of the *on* state predicate, because no more than  $n-1$  blocks can be stacked directly on top of other blocks. Also, the only possible values for the abstracted *holding* state predicate are zero and one, since the domain theory restricts resources to only one robot arm with which the blocks can be picked up. Effectively, this reduces the possible abstract state-space to 4,840 possible states. Some of the remaining abstract states are still not possible. For example, abstract state that has a nine as a value of *on* state predicate dimension, cannot have more than a value of one in either *clear* or *on-table* dimensions (assuming ten object instances). The incremental construction of the case library assures that the number of both abstract and specialized states actually stored is minimal.

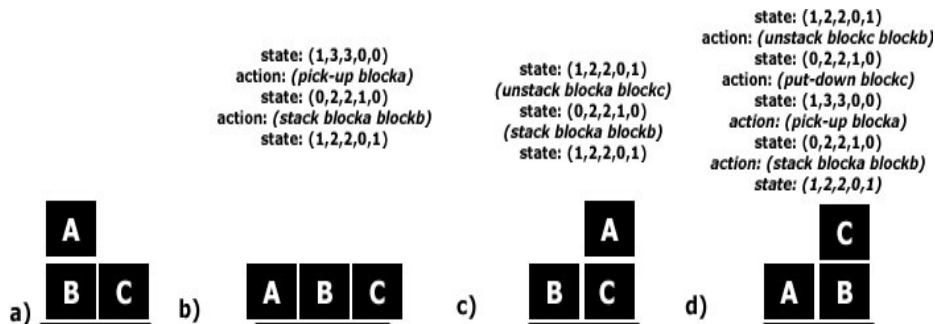


Fig. 4. a) The goal state reached by all of the problems. b-d) The initial states and abstracted plans, for problems  $p1$  (b),  $p2$  (c), and  $p3$  (d).

## 5. Implementation

To illustrate the approach discussed above, we will discuss two examples of the plan recognition task, implemented in the blocksworld planning domain. The observed planning agent is the PRODIGY nonlinear state-space planner whose execution cycle was slightly modified in order to observe the world states traversed along the solution path. Plans for problems successfully solved by the planner are then augmented to contain the state information and are passed to the recognizer for observation of simulated plan execution.

We first show how the knowledge about the world states can enable the recognizer to cope with situations in which a completely new action is encountered. Consider the problems in figure 4 where the planner's partial goal is to have blockA

on top of blockB. We start with an empty plan library, observe the very first plan ( $p1$ ), and store this plan in the case library. The contents of the library after the first plan is observed are depicted in figure 5. During observations of the second plan ( $p2$ ), the first action performed by the planner is *unstack*, which has not been seen before. However, the world state reached after *unstack* action was already explored in the past (the second state in plan  $p1$ ). The recognizer makes predictions based on the planner's decisions in the matched situation, and is able to correctly predict the next world state and planner's goal as depicted in figure 6a. The same scenario happens when the recognizer observes plan  $p3$ , because action *put-down* is seen for the first time. Again, the recognizer forms correct predictions based on the decisions made in a similar situation (the first world state in plan  $p1$ ) as depicted in figure 6b.

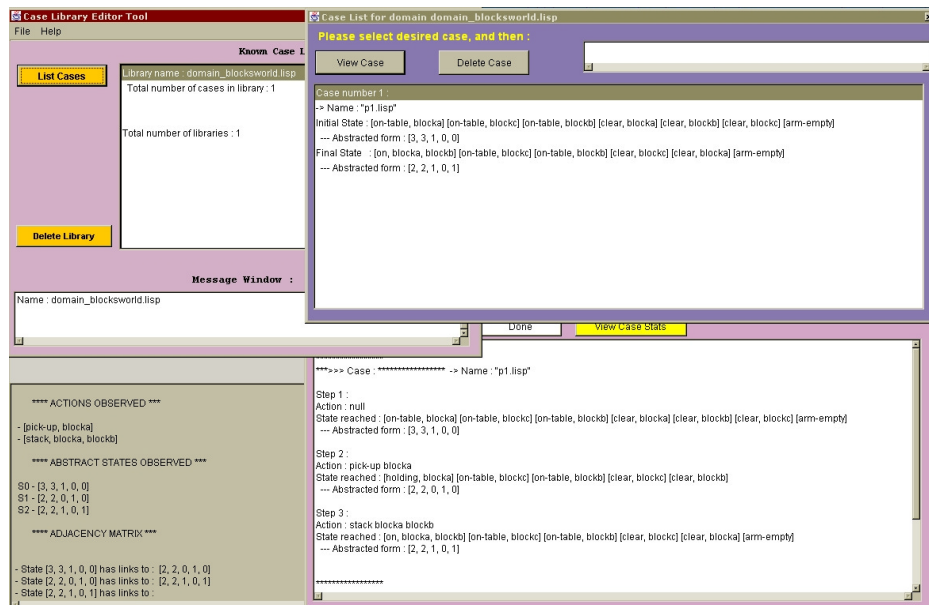


Fig. 5. The contents of the plan library after a single plan  $p1$  is observed.

The case-based nature of the plan recognition system enables it to form predictions even in completely new situations, because the situations from past plans (cases) can facilitate partial matches of those cases with the current situation. To illustrate this point, assume that the only plans observed by the recognizer are the plans described in the previous example and the initial world state of a new plan ( $p4$  of figure 1), having all three blocks stacked in a tower) has just been observed. The recognizer first abstracts the observed world state into its abstract state vector representation of (1,1,1,0,2), and attempts to find cases indexed by this abstract state in its library. As this abstract state is a completely new one, this first retrieval phase finds no matches to the current abstracted situation. The recognizer next searches its state-space graph for neighbor states, and successfully finds the abstract state (1,2,2,0,1) at the shortest distance from the current abstract state. Three states are indexed by the matched abstract state, and one of these states is the goal state. The matched goal state

is eliminated, because the planner made no subsequent decisions after this state was observed. At this point, the recognizer is able to use two specialized states to

```

...
Observing the next plan step...
- Action observed : [unstack, blocka, blockc]
- Current state :
[[holding, blocka], [on-table, blockc], [on-table, blockb], [clear, blockc], [clear, blockb]]
***** Abstracted state : [2, 2, 0, 1, 0]

Retrieve_Matches : EXACT MATCHES AT ABSTRACT STATE LEVEL !!!!
For state [2, 2, 1, 0, 1] following cases were found : [PlanRecognition.Case@1ba640]
...
Observing the next plan step...
- Action observed : [stack, blocka, blockb]
- Current state :
[[on, blocka, blockb], [on-table, blockc], [on-table, blockb], [clear, blockc], [clear, blocka], [arm-empty]]
***** Abstracted state : [2, 2, 1, 0, 1]

Verifying the predictions made at the previous planning step :

Predicted next state correctly : true, predicted state was :
[on, blocka, blockb] [on-table, blockc] [on-table, blockb] [clear, blockc] [clear, blocka] [arm-empty]
--- Abstracted form : [2, 2, 1, 0, 1], Predicted abstract state was [2, 2, 1, 0, 1]

Predicted next action correctly : true, predicted action was "stack blocka blockb"

6a)

...
Observing the next plan step...
- Action observed : [put-down, blockc]
- Current state :
[[on-table, blockc], [on-table, blocka], [on-table, blockb], [clear, blockb], [clear, blocka], [clear, blockc], [arm-empty]]
***** Abstracted state : [3, 3, 1, 0, 0]

Retrieve_Matches : EXACT MATCHES AT ABSTRACT STATE LEVEL !!!!
For state [3, 3, 1, 0, 0] following cases were found : [PlanRecognition.Case@1ba640]
Adding an edge into state-space graph...
...
Observing the next plan step...
- Action observed : [pick-up, blocka]
- Current state :
[[holding, blocka], [on-table, blockc], [on-table, blockb], [clear, blockb], [clear, blockc]]
***** Abstracted state : [2, 2, 0, 1, 0]

Verifying the predictions made at the previous planning step :

Predicted next state correctly : true, predicted state was :
[holding, blocka] [on-table, blockc] [on-table, blockb] [clear, blockc] [clear, blockb]
--- Abstracted form : [2, 2, 0, 1, 0], Predicted abstract state was [2, 2, 0, 1, 0]

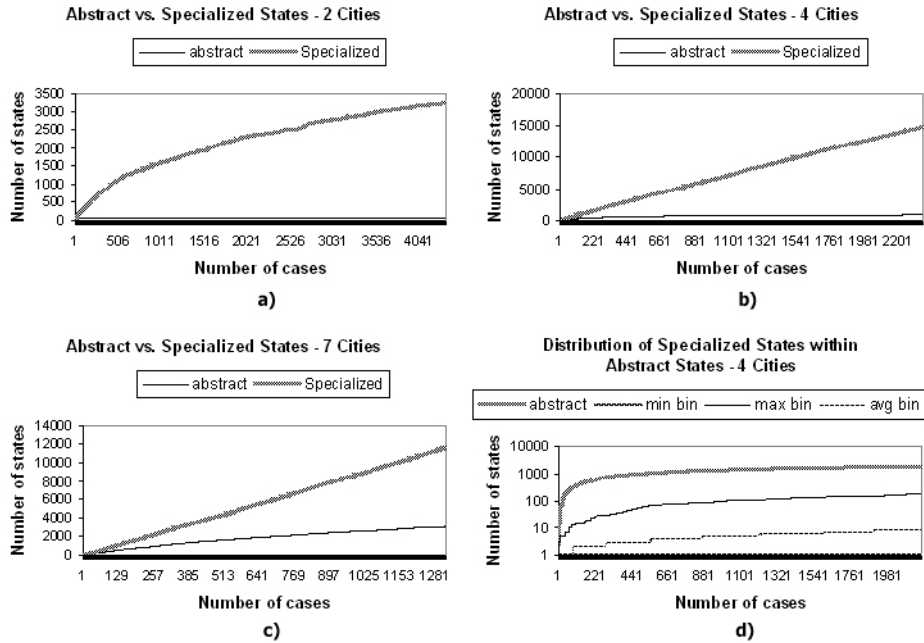
Predicted next action correctly : true, predicted action was "pick-up blocka"

6b)

```

**Fig. 6.** The predictions made by the recognizer in light of unknown actions. a) Predictions after action *unstack* in plan p2 is observed for the first time; b) Predictions after a new action *put-down* in plan p3 is observed.

form predictions. As both matched states were followed by an *unstack* action, the recognizer successfully predicts planner's next action of unstacking blockC off of the block it is on. The goal prediction in this particular case is trivial, given a match and only one goal state observed. The recognizer then observes the next plan step consisting of an *unstack* action and abstract state (0,1,1,1,1), also seen for the first time. Nevertheless, predictions can be formed again, in the same fashion as done before.



**Fig. 7.** a) – c) represent the number of abstract versus specialized world states in the logistics planning domain for 2, 4, and 7 cities respectively. d) is a logarithmic plot of the distribution of specialized world states indexed by their abstract state “bins”.

In order to experimentally determine the number of possible abstract world states for a given domain, we randomly generated problem sets in the logistics planning domain. Each problems set contained 3000 problems and a fixed number of cities. The number of cities ranged from 2 to 9, for a total of 8 problem sets. We executed the problems using PRODIGY, with a time-bound of 16 seconds. The planner successfully generated solutions (plans) to about 80% of the problems (some did not finish in the time allowed). We then considered the generated plans as inputs to the plan recognition system in order to determine the ratio of abstract states versus the specialized states, as well as the distribution of specialized states into “bins” indexed by abstract states. As we can see from figure 7<sup>7</sup>, the number of abstract states observed is much smaller than the number of specialized states, and the rate at which new abstract states are recognized is also considerably smaller than the recognition rate of abstract world states. Figure 7(d) shows a logarithmic plot of the number of abstract states (first-level indexes) and the statistics for the specialized states which they index. The minimum (1) and the maximum (201) bin sizes represent extremes, while the average bin size reaches 16 specialized states for the plans collected. These results are very encouraging. Our future research efforts will experimentally determine the asymptotic abstract state-space behavior for other planning domains.

<sup>7</sup> Due to a lack of space, not all of the results are depicted in figure 7.

## 6. Conclusion

This paper has introduced a novel method for performing keyhole plan recognition using case-based reasoning. Rather than requiring a hand- (or otherwise) generated case library in advance, we have shown a method for building the library incrementally. This solves a number of problems associated with many plan recognition algorithms. These problems include the assumption of a complete case-base and the danger of irrelevant cases. By incrementally adding plans to the case library as they are observed, we avoid the inclusion of theoretically possible cases that do not in practice occur. We have also shown how plan recognition can take place even in the face of novel observed behavior.

Furthermore, we have provided a novel indexing scheme and similarity metric that uses abstract states as indices. The use of abstract states rather than an exhaustive set of ground literals provides a tractable means for retrieval when a large set of cases exist. Further research will produce empirical results to demonstrate the behavior of the system in large case libraries.

## Acknowledgements

This paper is supported by the Dayton Area Graduate Studies Institute (DAGSI) under grant #HE-WSU-99-09, by a grant from the Information Technology Research Institute (ITRI), and by Wright State University (Research Challenge: Young Investigator grant). We thank the anonymous reviewers for their valuable comments.

## References

1. Allen, J. F., and Perrault, C. R.: Analyzing intention in dialogues. *Artificial Intelligence* 15(3) (1980) 143-- 178.
2. Albrecht, D. W., Zukerman, I, and Nicholson, A. E.: Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8 (1998) 5--47.
3. Blythe, J.: Planning under uncertainty. *Doctoral thesis. Technical Report, CMU-CS-98-147*. Computer Science Dept., Carnegie Mellon University (1998).
4. Bauer, M.: Machine learning for user modeling and plan recognition. In V. Moustakis J. Herrmann, editor, *Proc. ICML'96 Workshop "Machine Learning meets Human Computer Interaction"* (1996) 5--16.
5. Bares, M., Canamero, D., Delannoy, J.-F., and Kodratoff, Y.: *XPlans: Case-Based Reasoning for Plan Recognition*. *Applied Artificial Intelligence* 8 (1994) 617—643.
6. Carberry, S.: *Plan Recognition in Natural Language Dialogue*. MIT Press (1990).
7. Carbonell, J. G., Blythe, J., Etzioni, O., Gil, Y., Joseph, J., Kahn, D., Knoblock, C., Minton, S., Perez, A., Reilly, S., Veloso, M., and Wang, X.: *Prodigy4.0: The Manual and*

Tutorial. *Technical Report, CMU-CS-92-150*. Computer Science Dept., Carnegie Mellon University (1992).

8. Fikes, R. and Nilsson, N.: STRIPS: A new approach to the application of theorem proving to problem solving. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*. Morgan Kaufmann (1990).
9. Fish, D.: A Dynamic Memory Organization for Case-Based Reasoning Supporting Case Similarity Determination and Learning Via Local Clustering. *Masters Thesis*. Computer Science Dept., University of Connecticut (1995).
10. Hammond, C.: *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, San Diego (1989).
11. Kautz, H.: A formal theory of plan recognition and its implementation. In J. Allen, H. Kautz, R. Pelavin and J. Tenenber, *Reasoning about plans*, Morgan Kaufmann (1991).
12. Lesh, N., and Etzioni, E.: Scaling up goal recognition. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning* (1996) 178--189.
13. Penberthy, J. S., and Weld, D. S.: *UCPOP: A sound, complete, partial order planner for ADL*. In *Proceedings of KR-92* (1992) 103--114.
14. Veloso, M.: *Planning and learning by analogical reasoning*. Springer-Verlag (1994).
15. Veloso, M., Carbonell, J., Perez, A., Borrajo, D., Fink, E. and Blythe, J.: Integrating planning and learning: The PRODIGY architecture. *Journal of Theoretical and Experimental Artificial Intelligence*. 7(1) (1995) 81--120.
16. Veloso, M. M., Pollack, M. E., and Cox, M. T.: Rationale-based monitoring for continuous planning in dynamic environments. In R. Simmons, M. Veloso, & S. Smith (Eds.), *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*. Menlo Park, AAAI Press (1998) 171--177.
17. Weida, R. and Litman, D.: *Terminological plan reasoning and recognition*. In *Proceedings of the Third International Workshop on User Modeling* (1992) 177--191.