

Case-Based Plan Recognition with Incomplete Plan Libraries

Boris Kerkez and Michael T. Cox

Department of Computer Science and Engineering
 Wright State University, Dayton, OH
 {bkerkez,mcox}@cs.wright.edu

Introduction

The main focus of this research is to establish the techniques and prove feasibility of the case-based keyhole plan recognition dealing with incomplete plan libraries. Most traditional plan recognition systems operate with complete plan libraries that contain all of the possible plans the planner may pursue. However, enumeration of all possible plans may be difficult (or impossible) in some complex planning domains. Furthermore, the completeness of the library may result in occurrence of extraneous plans that may impact the recognizer's efficiency (Lesh and Etzioni, 1994).

The main difficulty when dealing with incomplete plan libraries is the recognizer's inability to reason about the planner's intentions that are not contained in the plan library. The recognizer that deals with incomplete plan libraries will only be useful if it is capable of making the intent predictions based on the limited information already present in the library. Most traditional recognition systems reason in terms of planning actions and do not explicitly keep track of the world states visited during the execution of a plan, except for the initial and the goal states. On the other hand, our system reasons in terms of both actions and situations in which the planner finds itself. Planning situations, represented by the states of the planner's environment, enable the recognizer to form predictions in cases where traditional systems would falter, such as making predictions in light of novel planning actions. Our experimental results demonstrate the effectiveness of the case-based plan recognition with incomplete libraries.

Plan Representation

The additional knowledge that enables the recognizer to reason in light of novel planning actions comes in the form of the intermediate planning states. Traditional state-space planners typically represent a plan with the initial state, the goal state, and a sequence of operators that transform the former state into the latter. We extend the plan representation to implicitly include the intermediate planning states visited on the path from the initial to the goal state. An example of the extended plan representation is shown in Figure 1. A plan is a sequence of action-state pairs, where each state is paired up with the action that

causes the transition to that particular state. The initial state is paired up with a *null* action for the sake of consistency.

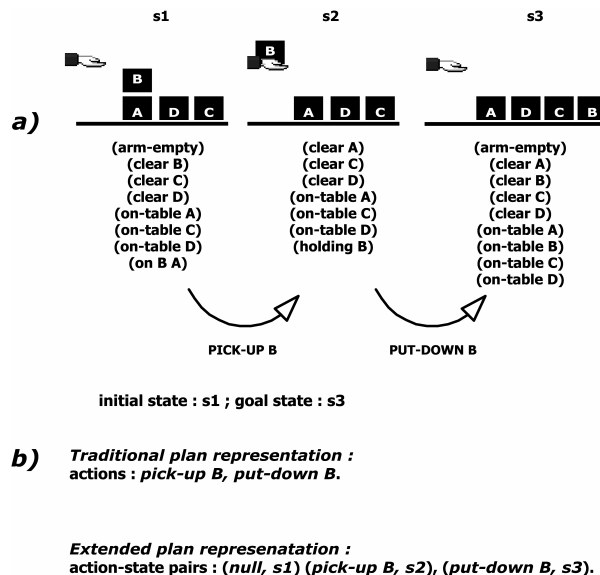


Figure 1. a) An example of a simple planning episode from the blocksworld planning domain. b) Two different views of the observed plan.

Most traditional plan recognition systems form predictions by matching the partial plans, formed from the observations of the planner's behavior, with the plans in the plan library. Incompleteness of libraries requires different prediction techniques, because plans, whose prefixes match the observed partial plans, may be absent from the library.

Our system tackles this issue by forming the intent predictions based on the planner's current world state. Each time the planner commits to an execution of an action, it transitions to a specific situation defined by the reached world state. Because all of previously visited situations are stored in the library as components of the extended plans, the recognizer has the ability to attempt to find previously visited situations that are similar to the current situation in hand. If such a past situation is found, it can be used to guide the current intent predictions by recalling previously made choices. When the recognizer encounters a novel planning action, it also uses the knowledge about the planner's current state of the world in order to retrieve similar past situations from the library and be able to make predictions even in light of newly observed planning

actions. In essence, by storing the intermediate state information, we now have the ability to split an observed partial plan $p = \{(a_1, s_1), \dots, (a_n, s_n)\}$ into any of the plans $\{(a_i, s_i), \dots, (a_n, s_n)\}$, where $i = \{1, \dots, n\}$.

An additional benefit of plans represented as sequences of action-state pairs is the ability to reason about plans with no explicitly defined initial states. Because our system keeps track of intermediate planning situations, any intermediate state can serve as the initial state of some plan. Such flexibility enables the recognizer to be applicable in continuous planning domains where there are no clear distinctions among the world states visited during the plan execution.

Given an appropriate similarity metric, the recognizer may be able to form predictions even in cases when no matches to observed partial plans of any length can be found. This is because the currently observed planning situation may be similar to some previously observed situation contained in the plan library. The prediction of the planner's current intent can then be guided by inspecting the planner's actions at the previous steps. Keeping track of all intermediate planning states increases the chances of finding past situations that are similar to the current situation in hand.

Abstraction and Indexing

Although the intermediate states are very helpful when recognizing plans with incomplete plan libraries, the state-space for a given planning domain may be quite large (Kerkez and Cox, 2001). A large number of possible situations negatively affect the retrieval efficiency of the recognizer. We developed an indexing and retrieval scheme based on the concept of state abstraction that allows the recognizer to reduce its search space by focusing only on relevant subsets of the state-space where potential similar past situations may be found. The abstraction scheme, as well as the whole recognition process, requires that the states in the planner's environment are represented as collection of ground literals, a trademark of state-space planners (Carbonell *et al.*, 1992). Another requirement is that the objects, which are the arguments of literals, have an associated abstraction hierarchy that allows efficient root type determinations. Figure 2 shows an example of a state from the blockworld domain and its abstract representation. The abstracted states are non-negative integer vectors, whose dimension values indicate the count of occurrences of literals of a particular root type.

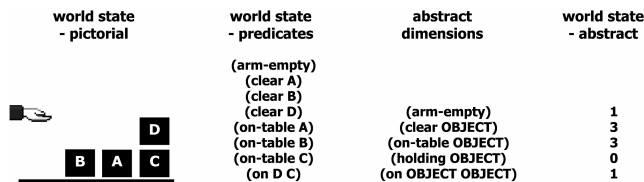


Figure 2. An example of representational scheme from the blockworld planning domain.

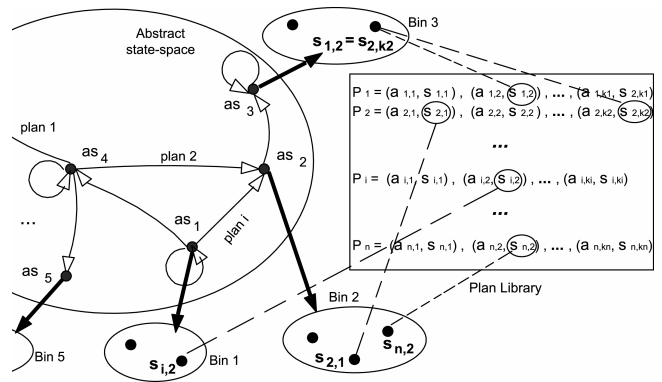


Figure 3. Indexing and storage structures. Abstract states (as_i) point to bins (bold lines), containing world states (s_j). World states in turn point (dashed lines) to past plans (P_j) in which they are contained.

The main indexing structures in the context of the case-based plan recognition are shown in Figure 3. Abstract states point to structures called bins, which contain the concrete world states. Each state in a single bin has an identical abstract representation, which allows for the efficient retrieval of past situations that are similar at the abstract level to the currently observed situation. Once the correct bin is located, the plans containing concrete states within the bin are retrieved and used to guide the current predictions of the planner's intent. Because the state-space size can be large for complex planning domains, our system constructs the plan library incrementally from the observations of the planner's behavior. Such incremental construction also minimizes the occurrence of extraneous plans, because only the plans pursued by the planner are actually stored in the library.

An issue that arises with abstract indexing is the potential saturation of bins. That is, some bins may contain a large number of concrete states, which in turn may point to a large number of past plans, and may influence the recognizer's efficiency. Given an equivalence relation on the set of the concrete world states, the bins may be further divided into disjoint subsets that further narrow the search space. To achieve this, we utilize a representation changing technique that transforms the world states into corresponding *state graphs* (Kerkez, 2002). An equivalence relation based on the isomorphism mapping among the state graphs provides a means to further divide the states in a single bin so that the states that are structurally identical are in the same equivalence class. However, graph isomorphism is computationally intractable for all but the simplest planning domains. We developed a sub-optimal pseudo-isomorphism equivalence relation whose equality comparison time is linear in the number of state literals (Kerkez and Cox, 2002). We have also shown that the accuracy of the local action prediction increases significantly with the pseudo-isomorphism equivalence relations.

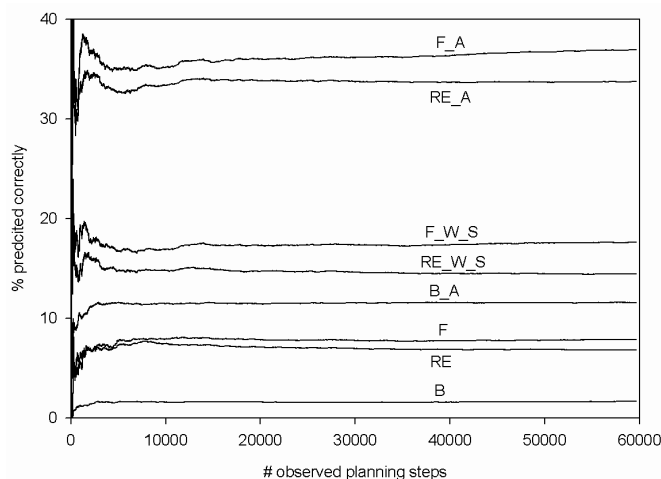


Figure 4. Percentages of correctly predicted next actions, abstract (suffix “_A”) and concrete, with (suffix “_W_S”) and without argument substitutions, for the 3-city logistics domain.

Current implementation of our system works with the PRODIGY state-space planner, whose execution cycle has been amended to include the intermediate state information. We experimentally evaluated the recognition performance for predictions that are local to the currently observed planning state, consisting of recognizing the next planning action. Figure 4 shows the percentages of correctly predicted next actions for eight different action selection strategies in the 3-city logistics domain while observing about 60,000 planning steps. The recognizer first predicts the next action type without predicting the action arguments. Such predictions are called abstract level predictions and are indicated in the figure by the suffix “_A”. After choosing the action prediction at the abstract level, the recognizer may reuse the arguments from the chosen past action, or it may attempt to substitute the past action arguments with their counterparts with respect to the current situation in hand. Strategies involving the argument substitutions are indicated by the suffix “_W_S” in Figure 4.

Baseline (*B*) tests consist of choosing an action at random from a pool of all previously observed actions. Random elimination (*RE*) strategy chooses an action at random from an equivalence class in a single bin that matches the abstract representation of the currently observed situation, while the most frequent (*F*) strategies choose the action pursued with the highest frequency among the potential candidates. We can see from Figure 4 that *RE* strategies significantly outperform the baseline strategies, while *F* strategies perform slightly better than *RE* strategies. Concrete action predictions with argument substitutions perform significantly better than their counterparts without the argument substitutions. Further research efforts will concentrate on improving the local prediction accuracy with the help of various heuristics, as well as making the global predictions concerning the goals and the plans of the planning agent.

Conclusion

Given the low-level knowledge intensity, the data-driven recognition approach, and the ability to reason in light of novel plans, the recognition system presented here is applicable to a wide variety of planning domains. Whenever the planner’s environment can be amended to display the plans as sequences of action-state pairs, and whenever state abstraction is possible, the system may benefit from the recognition techniques described in here. One potential future application of the recognizer is in the domain of computer-aided tutoring, where the planner is a human being trained to operate a computer system. In performing the actions on the screen by clicking the input devices, a human user changes the state of the environment in which he or she is currently working. Because human trainees have goals associated with the tasks they are performing, they are effectively performing sequences of state-changing actions that (hopefully) lead to the satisfaction of their goals. A plan recognition system may be utilized to recognize potential faulty user plans and subsequently tutor the user towards the correct solution. For this scenario to be feasible, the recognizer would have to possess the ability to monitor the actions performed by human users in terms of (typically) keyboard and mouse clicks. Our future research efforts will focus on establishing techniques to monitor computer user’s actions and applying the plan recognition techniques to the user interface planning domains.

References

- Carbonell, J. G., Blythe, J., Etzioni, O., Gil, Y., Joseph, R., Kahn, D., Knoblock, C., Minton, S., Perez, A., Reilly, S., Veloso, M., & Wang, X. (1992). PRODIGY 4.0: The Manual and Tutorial (Tech. Rep. No. CMU-CS-92-150). Carnegie Mellon University, Department of Computer Science, Pittsburgh, PA.
- Kerkez, B. (2002). Learning Plan Libraries for Case-based Plan Recognition. In *Proceedings of the 13th Midwest Artificial Intelligence and Cognitive Science Conference*. Illinois Institute of Technology, Chicago, IL.
- Kerkez, B., & Cox, M. (2001). Case-based plan recognition using state indices, In D. W. Aha, I. Watson, & Q. Yang (Eds.), *Case-based reasoning research and development: In Proceedings of 4th international conference on case-based reasoning* (pp. 227-242). Vancouver, Canada: Springer-Verlag.
- Kerkez, B., & Cox, M. (2002). Incremental Plan Recognition with Incomplete Plan Libraries, In *Proceedings of 1st European Conference on Case-Based Reasoning* (formerly EWCBR), Aberdeen, Scotland
- Lesh, N., & Etzioni, O. (1996). Scaling up goal recognition. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning* (pp 178-189).