A New Model for Automatic Generation of Plan Libraries for Plan Recognition

Martín G. Marchetta
Logistics Studies and Applications Centre.
Facultad de Ingeniería. Universidad Nacional de Cuyo.
Centro Universitario, CC 405 (5500), Mendoza, Argentina.
Tel: 0054-261-4135000 ext 2128
e-mail: mmarchetta@fing.uncu.edu.ar

Raymundo Q. Forradellas
Logistics Studies and Applications Centre.
Facultad de Ingeniería. Universidad Nacional de Cuyo.
Centro Universitario, CC 405 (5500), Mendoza, Argentina.
Tel: 0054-261-4135000 ext 2128
e-mail: kike@uncu.edu.ar

Abstract
In the context of Computer Aided Process Planning (CAPP), feature recognition as well as the generation of manufacturing process plans are very difficult problems. The selection of the best manufacturing process plan usually involves not only measurable factors, but also idiosyncrasies, preferences and the know-how of both the company and the manufacturing engineer. In this scenario, mixed-initiative techniques such as plan recognition, where both human users and intelligent agents interact proactively, are useful tools for improving engineer's productivity and quality of process plans. In order to be effective, these intelligent agents must learn autonomously this preferences and know-how. The problem of learning plan libraries for plan recognition has gained much importance in recent years, because of the dependence of the existing plan recognition techniques on them, and the difficulty of the problem. Even when there is considerable work related to the plan recognition process itself, less work has been done on the generation of such plan libraries. In this paper, we present some preliminary ideas for a new approach for acquiring hierarchical plan libraries automatically, based only on a few simple assumptions and with little given knowledge.

Keywords: Computer Aided Process Planning, Plan Recognition, Intelligent Agent, Plan Library Learning, Machine Learning, Planning

INTRODUCTION

The complexity of recognizing manufacturing features from a part design, along with the difficulty of generating a suitable (and in some sense good) manufacturing process plan has largely been recognized. One of the problems of feature recognition is that of multiple interpretations of CAD designs in terms of manufacturing features. Since the resulting process plans depends on the feature model generated, multiple feature models (and also multiple process plans) must be considered.

Moreover, in addition to quantitative factors (such as cost and time), the selection of the best part interpretation in terms of features and the best process plan, often involves non-measurable factors, such as preferences, company's culture and know-how of the manufacturing engineers. This situation is reflected in the fact that in practice, engineers in the industry prefer the use of interactive tools rather than totally automated ones (Horváth and Rudas, 1993; Horváth and Rudas, 1994; Horváth and Rudas, 1995; Gao, 1999).

These are the main reasons why practical CAPP systems may increase their effectiveness by supporting mixed-initiative approaches. In these approaches, autonomous intelligent agents assist engineers proactively, but always considering the constraints that these users impose interactively. Thus, intelligent assistance is always guided by the (sometimes partial) decisions of the system users.

In order to allow this kind of proactive intelligence, the identification of the engineer's intentions must be performed while he is working. Once these intentions have been inferred, the intelligent agent may provide different kinds of aids, such as improvements proposals on the manufacturing process design, and the performance of tasks on behalf of the user. Plan recognition is a useful technique for identifying user's intentions.

Most existing techniques for plan recognition are based on the use of a plan library previously created (Kautz, 1987; Kautz, 1991; Lesh et al., 1999; Goldman et al., 1999). This library contains the different plans the user may pursue when using the application (within this work, the application is a CAPP system). There are different representations for plans, but all of them basically represent a set of actions that must be performed, along with a set of constraints on their execution.

Until some years ago, these plan libraries were hand coded by a human expert, which is difficult, tedious, and it makes difficult to port plan recognition systems to new domains (for example to different industries).

In recent years, the automatic and semi-automatic generation of plan libraries has gained much importance, because of the reasons mentioned above. Some approaches, with different results, have been developed.

A known way for acquiring plan libraries automatically is to generate all possible goals and plans for a given domain, possibly using some bias to allow only valid goals and plans (Lesh et al., 1999; Lesh, 1998).

On the other hand, some works present an approach based on the acquisition of plan libraries from example cases. The generation of such libraries by this method can be achieved in several ways. One of them is to use some abstraction method to produce a non-hierarchical, subsuming plan for a set of action sequences that achieves a known goal, using labeled examples and some knowledge about an abstraction hierarchy of concepts (Bauer, 1998a; Bauer, 1998b). In (Bauer 1999a), a clustering approach is presented, that eliminates the need of goal annotation for examples. In (Bauer 1999b), a similar algorithm is presented, but the approach here is to group similar action sequences that belong to a known goal, thus generating alternative decompositions for that goal.

Another approach is to generate hierarchical decompositions of goals and composed actions, from labeled example cases (Garland et al., 2000; Garland et al., 2001).

In addition to the difficulty of porting plan recognition systems to new domains, an important reason for automating the generation of plan libraries is autonomy. The works mentioned above automate some of the tasks that are needed to build plan libraries. However, all of them require some human expert intervention in order to work. This is an important problem since the intelligent agents supporting manufacturing process planning should learn the company's know-how, and improve their own performance with experience.

As far as we know, the automatic acquisition of hierarchical plan libraries, from unlabeled example cases (that is, unsupervised learning of plan libraries), has not been achieved yet. In this paper, we present some ideas for acquiring such libraries, with little hand coded knowledge and unlabeled example cases. An example from the flexible packaging domain, as described in (Ibañez et al., 2003) and (Ibañez et al., 2001), is given to illustrate the algorithm.

PLAN LIBRARIES GENERATION ISSUES

There exist many representations for plan libraries, which contain different kinds of information. The algorithm presented in this paper is focused on the specific aspect of inferring plan decompositions from unlabeled action sequences.

A decomposition of an action is a set of simpler actions that have to be performed in order to accomplish it. A composed action can have several alternative decompositions, and each decomposition can be made up of either other composed actions or other basic ones (see Kautz, 1987; Kautz, 1991).

In order to learn such decompositions, an automatic segmentation mechanism must be provided. Such a mechanism must have the capability of identifying higher-level actions, by grouping the primitive ones that are observed from the user's interaction with the application, detecting recurrent patterns.

One difficult problem that arises, is that of interleaved plans. Previous works assume that each action sequence observed, contains only actions related to one goal (Bauer, 1998a; Bauer, 1998b), or that there may be actions for more than one goal, but the segmentation is not inferred, but given by the trainer, by means of labeled examples (Garland et al., 2000; Garland et al., 2001). This could be true if the observations (action sequences), are provided by an expert, or by somebody who is explicitly training the system. However it would be useful to use an algorithm that can operate even with interleaved plans and unlabeled examples, in such a way that the agent can learn autonomously while the engineer uses the system.

GENERATION OF PLAN LIBRARIES

The model used for plans representation in this work is similar to that described in (Kautz, 1987) and (Kautz, 1991). Under this representation, a plan library is a graph containing actions (also called events) as its nodes. In this work, we use terms "action" and "event" as synonyms. Actions are connected by 2 kinds of edges: thick grey arrows represent "is a" relations and thin black arrows represent "part of" relations. A special action called End Event, has a "is a" relation with all the "top-level" actions (actions that are not part of another one, and that are performed for their own purpose). A plan library defined in this way is a structure that represents the hierarchy of actions that are relevant to a domain.

Before introducing the segmentation algorithm, we present some assumptions. First, we assume that each action sequence is complete, that is, it contains at least all the actions necessary to achieve every goal present on it. For example, if the action sequence AS_1 contains actions for two interleaved plans P_1 and P_2, then at least it contains all the actions needed for achieving the goals of P_1 and P_2.

Second, if two different plans appear always together in the observations, then they may be considered as the same one. The third assumption is that, if two composed actions share some basic ones, the hierarchy generated may not be exactly the real one, but an equivalent hierarchy, as will be shown in following sections. These two facts are assumed since the final aim of this work is to produce a system whose behavior is as much faithful as possible with respect of the user's behavior, even when the internal plan library is not an exact representation of the reality.

Finally, we assume that information of pre-conditions and effects of the primitive actions that can be observed is available (information for planning).

Decompositions

The first problem that must be solved is to infer segmentation of actions. Segmentation can be defined as the grouping of sets of actions that must be performed, for achieving a more general one. Thus, the segmentation is the result of the identification of decompositions of non-primitive actions. A desirable property of a segmentation algorithm is it being incremental, so the previously made inferences are used in subsequent steps, instead of storing the processed example cases.

Intuitively, whenever a set of new actions appears in an action sequence, this set of actions can be considered as a decomposition for some new (and previously unknown)

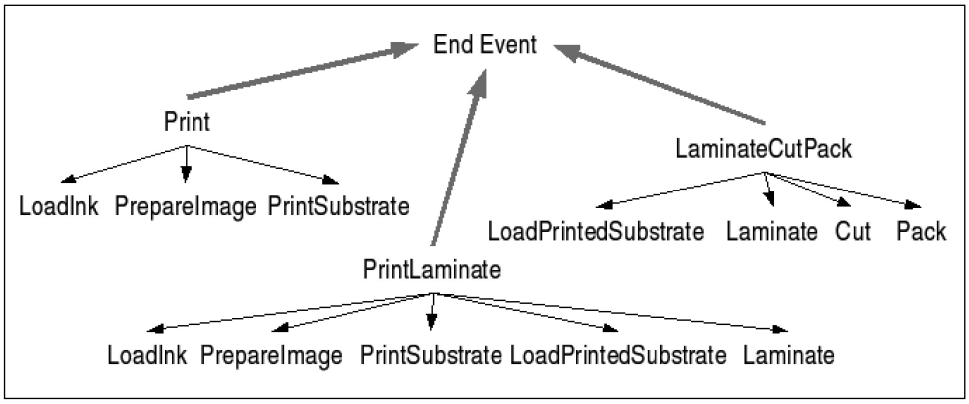

Figure 1: Simple plan library for packaging industry domain

non-primitive one that represents the observation. For example, consider the simple plan library presented in figure 1, which represents a simple plan library within the flexible packaging domain (see Ibañez et al., 2003).

Initially the system has an empty library that contains only the End Event. Now suppose that the agent perceives the following action sequence included by the user into a process plan during a session in the system:

$$AS_1 = \{LoadInk, PrepareImage, PrintSubstrate\}$$

Because none of the actions have been seen before, the agent creates a new composed action, named Print' whose parts are LoadInk, PrepareImage and PrintSubstrate (we denote the learned composed action with an apostrophe to show that the system will assign an arbitrary name to it).

A different situation arises when the agent observes an action sequence that contains parts of both known and unknown non-primitive actions. In this case, these actions can

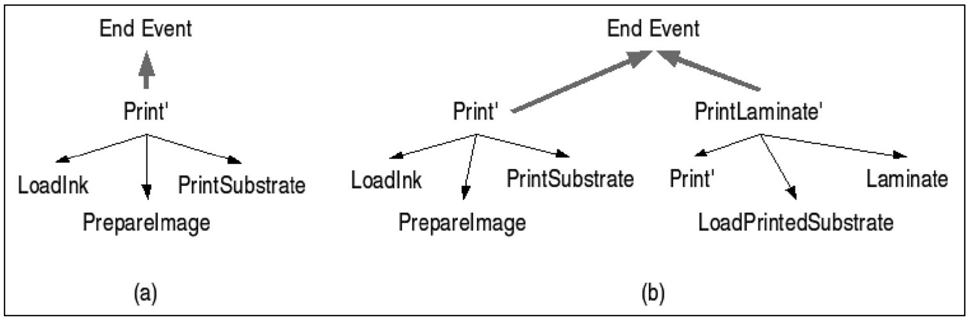

Figure 2: Plan library under construction

be considered as the parts of a more general event that represents the entire sequence. For example, suppose that the user includes the following actions in the process plan:

AS_2 = {LoadInk, PrepareImage, PrintSubstrate,LoadPrintedSubstrate,Laminate}

Here, LoadInk, PrepareImage and PrintSubstrate are identified as a decomposition of Print', so the algorithm will create a new composed action named PrintLaminate', which has Print', LoadPrintedSubstrate and Laminate as its parts. In figures 2.a and 2.b, the states of the plan library after the first and second observation, are shown.

Now suppose that the action sequences were seen in the inverse order. After the first sequence, the agent introduces in the plan library the action PrintLaminate', with LoadInk, PrepareImage, PrintSubstrate, LoadPrintedSubstrate and Laminate as its parts. When the second sequence is processed, only a part of a known action is identified. In these cases, the agent must figure out that, in fact, LoadInk, PrepareImage and PrintSubstrate are part of Print', which in turn is part of PrintLaminate'.

Thus, the action taken by the agent should be to create Print', and modify PrintLaminate' as needed. Finally, the new action is added to the plan library, in accordance with the segmentation, obtaining as before the state shown in figure 2.b.

Definition 1 (Complete decomposition).
A set of actions AS containing all of the parts of a composed action A_c,
is a complete decomposition of A_c.

Definition 2 (Partial decomposition).
A set of actions AS containing some parts of A_c, but not all of them,
is a partial decomposition of A_c.

The base of the method intuitively described before, is the identification of unknown actions, complete decompositions and partial decompositions. In absence of further information, all new action sequences are considered as part of a new non-primitive action. With the arrival of new example cases, the plan library is refined successively. These refinements are based on the complete and partial decompositions identified in the new observations.

A schematic algorithm for generating hierarchical decompositions for plan libraries, named HIDEL (HIerarchical DEcomposition Learner), is shown in Figure 3. This algorithm handles several cases that can arise, related to a new action sequence:

a. It has no previously seen basic actions.

b. It includes only complete decompositions of already known composed actions.

c. It presents some basic actions that have not been seen, and some actions that are a complete decomposition of already known actions.

d. It contains a partial decomposition of known actions.

```
HIDEL(AS , PLibrary)
Variables:
  AS: A new action sequence
  PLibrary: The current plan library
BEGIN
  CompDec = IDENTIFYCOMPLETEDECOMPOSITIONS(AS)
  PartDec = IDENTIFYPARTIALDECOMPOSITIONS(AS)
  Unknown = IDENTIFYUNKNOWNACTIONS(AS)

  NewSeq = CREATEACTIONSEQUENCE()
  for each decomposition in CompDec
     add composed action corresponding to decomposition to NewSeq

  add actions( Unknown) to parts(NewSeq)

  for each decomposition in PartDec
     NewAction = CREATEACTION()
     add actions(decomposition) to parts(NewAction)
     add NewAction to NewSeq
     replace actions(decomposition) with NewAction in PLibrary

  if empty(IDENTIFYCOMPLETEDECOMPOSITIONS (NewSeq))
     if elementCount(NewSeq) > 1
         NewAction = CREATEACTION ()
         add actions(NewSeq) to parts(NewAction)
         add NewAction to Plibrary as End Event
  else
     NewAction = getAction(1,NewSeq)
     If not(isSpecialization(End Event, NewAction))
         add NewAction to PLibrary as End Event
END HIDEL
```

Figure 3: Schematic segmentation algorithm

In case (a), the system can segment all the actions as part of a new composed event. In case (b), known non-primitive actions are identified from the sequence. Two situations may arise for this case. If the identified non-primitive actions constitute a complete decomposition, then the plan library already contains the information implied by the observation. For example, if the current state of the plan library is that of figure 2.b, the sequence

AS = {LoadInk, PrepareImage, PrintSubstrate,LoadPrintedSubstrate,Laminate}

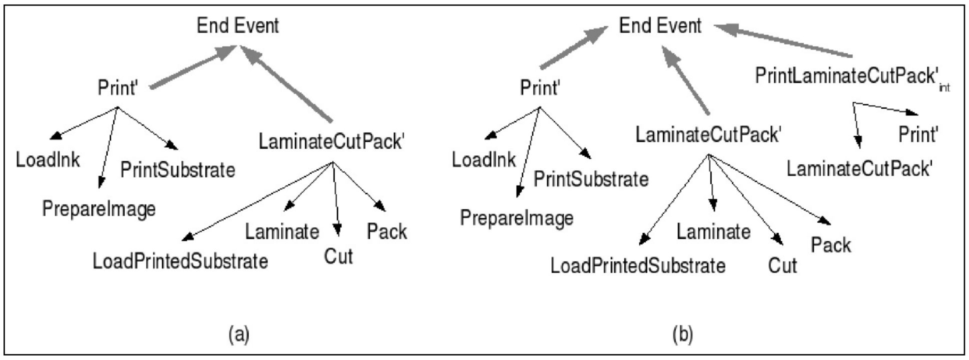

Figure 4: Plan library learned in case (b)

represents PrintLaminate′, so no change is needed for the plan library.

On the other hand, if the identified composed actions are not a complete decomposition, they are arranged as parts of a new action. Consider figure 4.a. After the sequence

AS = {LoadInk, PrepareImage, PrintSubstrate,LoadPrintedSubstrate,Laminate, Cut, Pack}

is observed, all the actions included in the observation are part of known composed actions Print′ and LaminateCutPack′). However, LaminateCutPack′ and Print′ are not a complete decomposition, so the system creates a new composed action PrintLaminateCutPack′$_{int}$, with them as its parts. The results of the segmentation are depicted in figure 4.b. The action PrintLaminateCutPack′$_{int}$ could either be a real plan (or subplan), or two interleaved plans (as in the example). Interleaved plans will be described in following sections.

In case (c) the algorithm builds a new composed event, whose parts are the unknown actions in the sequence and the composed actions identified from the known primitive ones observed. An example of this situation is the first one presented intuitively, where the following sequences were processed (see figure 2).

AS_1 = {LoadInk, PrepareImage, PrintSubstrate}
AS_2 = {LoadInk, PrepareImage, PrintSubstrate,LoadPrintedSubstrate,Laminate}

Finally, when the new observation contains a partial decomposition, as in case (d), it is likely that these actions are in fact parts of a composed event, which in turn is shared by the previously known one, and a new one that must be created. An example of this case, is the same as the previous one, with the observations in the inverse order:

AS_1 = {LoadInk, PrepareImage, PrintSubstrate,LoadPrintedSubstrate,Laminate}
AS_2 = {LoadInk, PrepareImage, PrintSubstrate}

Figure 5.a shows the state of the plan library after AS_1, and figure 5.b shows the state after AS_2. Here, actions LoadInk, PrepareImage and PrintSubstrate are grouped as part of Print', which is the new composed action created in accordance with the new observation, and PrintLaminate' is modified as needed.

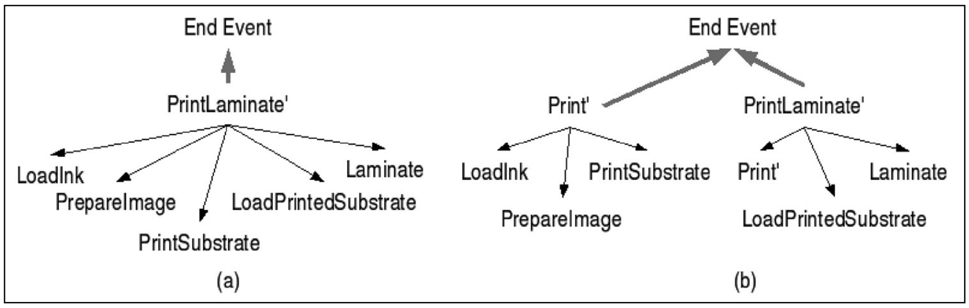

Figure 5: Plan library learned in case (d)

Interleaved plans

When an action sequence contains actions of more than one plan, some special problems arise. Interleaved plans generate difficulties because in absence of further information, a plan library learning algorithm cannot distinguish between interleaved plans, and an individual composed plan. Consider again the plan library shown in figure 1. Suppose that the action sequence

AS_1 = {LoadInk, PrepareImage, PrintSubstrate,LoadPrintedSubstrate,Laminate, Cut, Pack} is observed. Without additional information, it is difficult to infer if the whole observation corresponds to a single plan, or if it corresponds to two interleaved plans (Print and LaminateCutPack), so the system creates InterleavedPlans' to represent AS_1.

If the sequence

$$AS_2 = \{\text{LoadInk, PrepareImage, PrintSubstrate}\}$$

is processed, a new action is created whose parts are LoadInk, PrepareImage and PrintSubstrate, and InterleavedPlans' is updated. As can be seen, the new observation did not solve the problem.

We propose two alternatives to address this problem, depending on the bias imposed to the plan library representation. In (Kautz, 1987), an event can be a part of another one, or can be an End Event, but not both at the same time. If this bias is imposed to the representation of plan libraries, when an action that is part of a more general one appears alone in a new observation, as an end event (i.e, it appears as executed for its own sake), it follows immediately that the previously inferred composed action was, in fact, an interleaving of more than one plan. In the above example, when AS_2 is processed the composed action Print' is created, and because it appears as executed for its own sake, it follows that InterleavedPlans' contained, at least, two interleaved plans (Print' and the rest of InterleavedPlans'), so a reasonable modification to the plan library would be to remove actions of Print' from InterleavedPlans'.

However, in some domains it could be useful to allow an action to be a part of another one, and a top-level action at the same time (see Goldman et al., 1999). For these cases, another mechanism is required in order to avoid the ambiguity of interleaved plans. One feasible technique for this, is the use of a probabilistic rule. In (Marchetta and Forradellas, 2006), an algorithm was presented that follows this approach.
The solution proposed in that work is to compute the probability that some action A' is in fact the interleaving of several plans, instead of a real composed one. Let $N(A'_T)$ be the number of action sequences observed by the agent, in which either A' or any of its compounding parts appear. Then

$$N(A'_T) = N(A'_{\text{int}}) + N(A'_{\neg\,\text{int}}) \qquad [1]$$

where $N(A'_{\text{int}})$ is the number of action sequences where A' appears as an interleaved plan, and $N(A'_{-\text{int}})$ is the number of sequences where it appears as a real plan.

Since the agent cannot directly determine whether A′ or not an interleaved plan, the actual value of N($A'_{\text{-int}}$) is approximated as the number of sequences where any of the parts of A′ appear as end events.

Thus, the probability of A′ being an interleaved plan is computed as follows

$$P(A'_{\text{int}}) = \frac{N(A'_{\text{int}})}{N(A'_T)} = \frac{N(A'_{1e}) + N(A'_{2e}) + \cdots + N(A'_{ne})}{N(A'_T)} = \\ = P(A'_{1e}) + P(A'_{2e}) + \cdots + P(A'_{ne}) \qquad [2]$$

where N(A'_{ie}) is the number of sequences where the i-th part of A′ appears as an end event.

This probability can be used in two ways:

1. Comparing it with the probability that the action is real
2. Using a threshold, beyond which the action can be considered as an interleaving of plans

Shared steps

At first sight, the fact that two composed actions share some steps should not be a problem. However, there exist some situations in which the algorithm shown in figure 3 generates an incorrect plan library.

Consider again the plan library shown in figure 1. Suppose that the sequences

AS_1 = {LoadInk, PrepareImage, PrintSubstrate}
AS_2 = {LoadInk, PrepareImage, PrintSubstrate,LoadPrintedSubstrate,Laminate}
AS_3 = {LoadInk, PrepareImage, PrintSubstrate,LoadPrintedSubstrate,Laminate, Cut, Pack}

are processed. In figure 6, the plan library state after the processing of each sequence according to the schematic segmentation algorithm is shown.

In the example, InterleavedPlans′ represents the interleaving of the events PrintLaminate and LaminateCutPack. If the learning mechanism can handle interleaved plans (for example, using some of the approaches mentioned in the previous section), as new example cases are processed PrintLaminate′ will be eventually removed from InterleavedPlans′. This last modification should turn InterleavedPlans′ into a representation of LaminateCutPack′, but instead it results in a new action whose parts

are Cut and Pack, and which is not a real or significative one. Thus, the resulting plan library definition is not equal (or even equivalent) to that presented in figure 1
This problem arises when the following conditions hold:

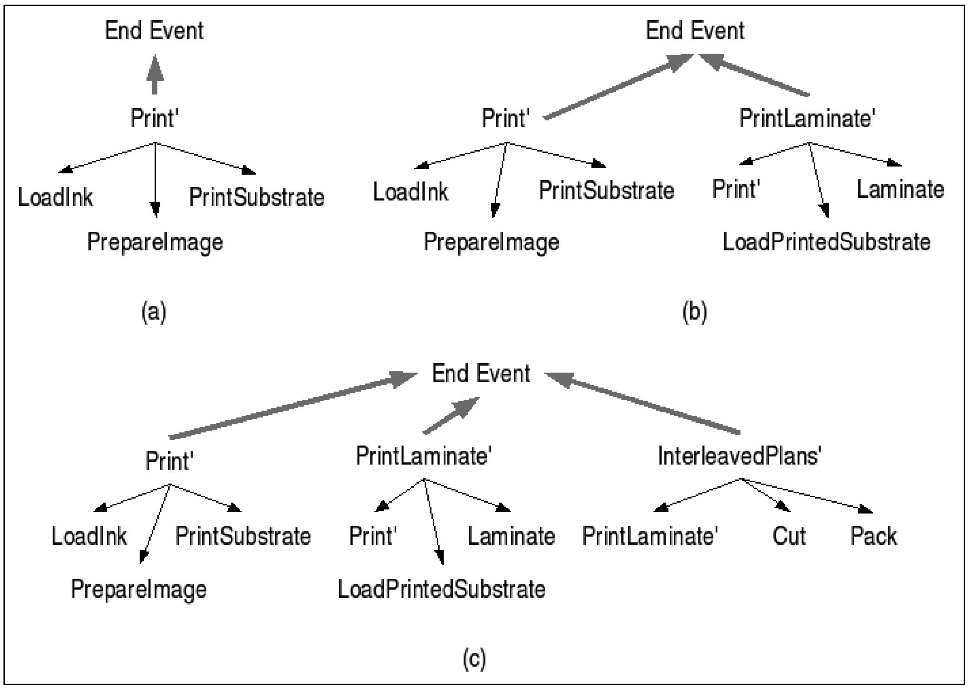

Figure 6: Plan library evolution for shared steps

1. Two composed actions shares some common parts
2. The composition of one of them is already known from previous observations
3. The other action appears later in an observation

Some modification could be made to the segmentation algorithm in order to avoid this situation, but we think that the most feasible solution is to define some mechanism to revise the definition of actions that could potentially share common steps, as new evidence is processed.

The algorithm presented in (Marchetta and Forradellas, 2006) for handling interleaved plans, which was briefly shown in the previous section, may solve this problem. In the proposed mechanism, with each new action sequence perceived by the intelligent agent, it computes for each non-primitive action of the plan library the probability that it is a real plan or an interleaving of two of them. When this probability reaches some threshold, the corresponding non-primitive action is removed from the plan library.

Since the problem presented in this section may lead to the introduction of some spurious non-primitive actions in the plan library, which will not appear systematically, the same mechanism may be used to remove them, and also to remove other kind of spurious actions from the plan library.

Alternative decompositions

Alternative decompositions of a non-primitive action, are the different ways in which it can be accomplished. Recall that a composed action is one that is accomplished when all its parts are performed. This type of actions may be accomplished in different ways. Within the decomposition of an action, its parts have an implicit conjunction relationship. Alternative decompositions, instead, have a disjunction relationship with each other within a composed action.

The conjunction of the effects of primitive actions that are part of some decomposition, can be seen as the goals that are achieved by that decomposition. If two actions sets are alternative decompositions of a composed action, then they both achieve the goal of the composed action. Thus, if two decompositions have the same conjunction of effects, then they can be considered as alternative decompositions of the same action.

Since we assume that information about the pre-conditions and effects of primitive actions is available as domain knowledge, alternative decompositions of actions as well as the goals of composed actions inferred, can be determined.

CONCLUSIONS AND FUTURE WORK

An algorithm for learning hierarchical plan libraries for plan recognition from unlabeled example cases was presented in this paper. Some complementary ideas for supporting interleaved plans and alternative decompositions were also introduced.

One of the long term objectives of this work, is to produce new techniques that allow a plan recognition system (part of an intelligent agent) to adapt automatically to new domains, assuming that known and new domains are modeled by the same basic actions (the actions used by the agent for planning), thus providing a high autonomy level to the agent using these techniques. In the CAPP domain, different companies may have the same manufacturing resources, but also have different preferences, policies and strategic objectives, which yield different manufacturing process plans. An intelligent agent with learning capabilities within a mixed-initiative environment may adapt itself to these differences.

Even when the ideas presented in this paper are promising, future work must include further development, in addition to an implementation and validation of the algorithms with real data from the CAPP domain within different industries. Some additional features must also be taken into account: abstractions of actions ("is a" relations), the inferring of actions orderings, actions with parameters, the inferring of restrictions on these parameters, and support of repeated and spurious actions.

REFERENCES

Bauer, M. (1998) "Acquisition of abstract plan descriptions for plan recognition", Proceedings of the fifteenth national/tenth conference on Artificial intelligence/ Innovative applications of artificial intelligence.

Bauer, M (1998) "Towards the Automatic Acquisition of Plan Libraries", Proceedings of the 13th European Conference on Artificial Intelligence.

Bauer, M (1999) "From Interaction Data to Plan Libraries: A Clustering Approach", Proc. of the Sixteenth International Joint Conference on Artificial Intelligence.

Bauer, M (1999) "Generation of Alternative Decompositions for Plan Libraries", IJCAI'99 Workshop on Learning about Users.

Gao, J (1999) "A Market Survey of Industrial Requirements for Product Data Management and Manufacturing Planning Systems", Proc. of the IEEE International Symposium on Assembly and Task Planning. Porto, Portugal.

Garland, A.; Lesh, N.; Rich, C.; & Sidner, C.L (2000) "Learning Task Models for Collagen", Proceedings of the AAAI Fall Symposium.

Garland, A.; Lesh, N.; & Sidner, C (2001) "Learning Task Models for Collaborative Discourse", Proc. of Workshop on Adaptation in Dialogue Systems, NAACL.

Goldman, R.P.; Geib, C.W.; & Miller, C.A (1999) "New model of plan recognition", In Laskey K. B. and Prade H. editors, Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence.

Horváth, L. & Rudas I.J (1993) "A Machine Learning Based Approach to Manufacturing Process Planning", Proceedings of the International Symposium on Industrial Electronics, ISIE'93.

Horváth, L. & Rudas I.J (1994) "Human Computer Interactions At Decision Making And Knowledge Acquisition in Computer Aided Process Planning Systems", International Conference on Systems, Man and Cybernetics. 'Humans, Information and Technology'.

Horváth, L. & Rudas I.J (1995) "Modeling Human-Computer Interactions in Collaborative Design and Planning", International Conference on Systems, Man and Cybernetics. 'Intelligent Systems for the 21st century'.

Ibañez, F.; Diaz, D.; Forradellas, R.Q (2001) "Scheduling for flexible package production", Proceedings IEPM. Vol. 1, 385-400.

Ibañez, F.; Diaz, D.; Forradellas, R.Q (2003) "An Algorithm for Minimising Due Times Violations in Flexible Package Production Scheduling", Journal of Computer Science and Technology, Vol 3 No.2.

Kautz, H (1987) "A Formal Theory of Plan Recognition", Ph.D. Thesis, University of Rochester.

Kautz, H (1991) "A Formal Theory of Plan Recognition and its Implementation", In Kaufmann, M. (Ed), Reasoning about plans, pp 69-126.

Lesh, N. & Etzioni, O (1996) "Scaling up goal recognition", Proc. of the Fifth International

Conference on Principles of Knowledge Representation and Reasoning.

Lesh, N (1998) "Scalable and Adaptive Goal Recognition", Ph.D. Thesis, University of Washington.

Lesh, N.; Rich, C.; & Sidner, C.L (1999) "Using Plan Recognition in Human-Computer Collaboration", Proceedings of the Seventh International Conference on User Modeling, pages 23–32.

Marchetta, M. & Forradellas, Q (2006) "Supporting Interleaved Plans in Learning Hierarchical Plan Libraries for Plan Recognition", Inteligencia Artificial Revista Iberoamericana de Inteligencia Artificial, Vol 10 No 32, pp 47-56.

Biography

Martín G. Marchetta is a PhD student at the Logistics Studies and Applications Centre at the National University of Cuyo, Argentina. He received his bachelor's degree in Information Systems Engineering from the National Technological University in 2002. His research interests include theoretical and applied artificial intelligence, business intelligence, software engineering, logistics and industrial systems.

Raymundo Q. Forradellas is a professor of Information Systems at the Industrial Engineering School (Engineering Faculty, at the National University of Cuyo), Director of the Masters in Logistics and Director of the Logistics Studies and Applications Centre. He received his PhD degree on Artificial Intelligence from the Polytechnic University of Madrid, Spain. His research interests include applied artificial intelligence systems, planning & scheduling, logistics and industrial systems.