SICS Technical Report T95:01
SICS-T--95/01-SE

ISSN: 1100-3154

A Simplistic Approach to Keyhole Plan Recognition

by

Annika Waern and Ola Stenborg

Swedish Institute of Computer Science
Box 1263, S-16429 Kista, SWEDEN

A Simplistic Approach to Keyhole Plan Recognition

Annika Wærn
annika@sics.se
Swedish Institute of Computer Science

Ola Stenborg
sten@sics.se
Stockholm University

January 4th, 1995

Abstract

When applying plan recognition to Human - Computer Interaction, one must cope with users exhibiting a large amount of reactive behaviour: users that change tasks, or change strategies for achieving tasks. Most current approaches to keyhole plan recognition do not address this problem. We describe an application domain for plan recognition, where users exhibit reactive rather than plan-based behaviour, and where existing approaches to plan recognition do not perform well.

In order to enable plan recognition in this domain, we have developed an extremely simplistic mechanism for keyhole plan recognition, "intention guessing". The algorithm is based on descriptions of *observable behaviour*, and is able to recognize certain instances of plan failures, suboptimal plans and erroneous actions. At run-time, the algorithm only keeps track of a limited number of the most recent actions, which makes the algorithm "forgetful". This property makes the algorithm suitable for domains where users frequently change strategies.

Keywords: Plan Recognition, Situated Reasoning

1 Introduction

Plan recognition is the task of inferring an agent's intentions (goal and plan for achieving that goal) from the the agent's observed actions. Plan recognition is an instance of a more general abductive explanation task, where a set of observations about the world are to be explained on the basis of some generic domain knowledge, if necessary by adding some assumptions about the state of the world.

The plan recognition problem appears in three different forms: Plan recognition when the actor is aware and actively cooperating to the recognition, for example by choosing actions that make the task easier (intended plan recognition), plan recognition when the actor is unaware of or indifferent to the plan recognition process (keyhole plan recognition), or plan recognition when the actor is aware of and actively obstructs the plan recognition process (obstructed plan recognition). Most research of today fall into one of the two first categories while the third generally is viewed as giving too little information and posing too hard meta-constraints on the recognition process to be useful to attack.

Although intended plan recognition *in principle* gets more information from the actor, the extraction of this information from actions and utterances is a complex task. The actors way of expressing him/herself must be interpreted - a system must for example know something about the actors language and domain knowledge, and it must also sometimes be able to reason about the actor's understanding of the systems competence to fully understand *how* the actor is trying to help the system understand his/her intentions. The complexity of the task is an effect of the richness of the information.

The fact that keyhole and obstructed plan recognition deal with less rich information sources imply that the recognition mechanisms for these tasks potentially could be simpler to realise, and this is indeed confirmed in existing literature. For example, the mechanisms proposed by Randall Calistri-Yeh [1991] for intended plan recognition in task-oriented dialogue are *very* complex, whereas most approaches to keyhole plan recognition of today base themselves on the simple plan structures from Kautz [986], without attempting to differentiate between the beliefs of the actor and those of the observing system. In order to deal with obstructed plan recognition, Ingrid Zukerman (personal communication) has proposed an even simpler recognition mechanism, where only typical collections of action occurrences are recognized and associated to goals, but where the plan recognition mechanism has the potential to *adopt over time* to change when the users ”catch on” to it.

In this paper, we argue that for many instances of keyhole plan recognition in human-computer interaction, even the plan structures from Kautz are unnecessarily complex. This is due to the fact that users in a keyhole plan recognition domain will exhibit *reactive* rather than cooperative behaviour.

2 Human plans and human behaviour

There are several applications of keyhole plan recognition. If the user's task is recognized, the system can inform the user if the goal is for some reason impossible to achieve, or help the user in learning a better way to achieving it by critiquing the user's actions, or even invoke a procedure to achieve it automatically. If the user's strategy (plan) for achieving the task is recognized (which allows the system to predict the user's future actions), the system can execute faster, or prevent incorrect or suboptimal behaviour. Finally, the user's strategy or task may play a role in collecting information about the user's knowledge of or preferences in the domain where the plans are executed.

However, when keyhole plan recognition is used to detect user plans in human-computer interaction, one must be aware of the structure of such plans. The nature of human plans was studied by Lucy Suchman [1987], work that has greatly influenced recent developments in reactive planning. Suchman claims that typically, humans *do* plan their behaviour, but foremost at a rather high and strategic level. At the level of individual interactions with the world, actions are selected reactively, based on the immediately perceived status of the world.

Not *all* human behaviour follow this pattern. A counter- example is task-oriented dialogues, where a dialogue usually deals with one singular task, and where the introduction of a new task or a modification of the current plan needs to be signalled [Litman and Allen 1987]. However, task-oriented dialogues arise from the need to perform *cooperative planning.* All agents in the dialogue are roughly committed to the overall task, and assume that the other agents are making inferences about the task, its sub-components and its results, as well as the commitments of the different agents.

In human - computer interaction, a user typically does not reason about the task as a joint task of her and the system. Instead, the user perceives the system as a *tool*, with which the user can perform a set of low-level manoeuvres or manipulations, which in different ways can be utilized to reach the user's overall goal. This attitude will cause users to change their intentions - both adopting a new plan for a goal, or even abandon the goal for a new one - without considering if the computer should know about the change. Furthermore, since the computer system is a low-level tool with respect to the user's goal, the user's singular actions may say very little about the user's overall task - many actions and action patterns will look the same no matter what the user's task is.

The general HCI situation is for this reason ill-formed for intended plan recognition, and indeed for most existing approaches to keyhole plan recognition as well. There exist applications where it is not correct to assume that a user keeps to one plan as long as her actions are possible to interpret within that plan as in [Kautz 1987], and neither is it always correct to assume that the user is maintaining *several* concurrent plans as in [Quast 1993] - a user may abandon an intention altogether. Furthermore, as our empirical study shows, we cannot be certain that the goals we would like to recognize in a particular application indeed will be recognizable. Before devising a plan recognizer for a particular application, we must analyse the application carefully to deduce *what* goals we can recognize, and how persistent users are in pursuing these goals.

3 A plan recognition task in news reading

We have made a minor study of an application domain where users exhibit a large amount of reactive behaviour. The study was done as a feasibility study for a project studying *news filtering* for usenet news [Karlgren et al. 1994], aimed at exploring whether plan recognition could provide a way to set up or adapt a news filter for an individual user. We had noticed previously in the project that the average user puts very little time in learning the functionalities of a news system, and was even less likely to be willing to configure a news filter manually. An alternative to manual configuration would be to let the user signal his or her preferences *while reading the news*. This way, the system can monitor the user's preferences, and correlate these to different properties of the entries (see [Kozierok and Maes 1993] for a similar example in a meeting booking system). We wanted to explore the possibilities for an even more "automatic" way of detecting preferences - if we could infer the user's interest in an entry directly from the reading pattern of the user, this could be used to gradually build up the filtering knowledge.

Our question was: Can we use keyhole plan recognition to detect enough of a users plan to find out if an entry is interesting or uninteresting to that user? Roughly, we were interested in recognizing:

- If the user searched for some particular information
- When the user succeeded in finding interesting information
- When the user's search for interesting information *failed* (detecting criteria on uninteresting information)

In addition to these tasks related to a specific information need, we found that the user often indulged in a "random exploration" task, without any specific information in mind.

3.1 Empirical studies of news readers

We selected to study the news reading task using a rather advanced news reading tool. The reason for choosing this tool was that it allowed a rather large graphical view of much of the information and used point-and-click interaction, and additionally it included functionalities of keyword search in headers. The assumption was that the rather high-level functionality of the tool would make keyhole plan recognition possible by exploiting a fairly close relation between what people actually do and their underlying intention.

The study was performed in two parts. First, we videotaped two users that were "thinking aloud" while reading. This study gave us a start at analysing the connection between different reading strategies, and the underlying reason for following a certain reading strategy. After analysing and classifying these, we made a second study of three "silent" interactions using new experiment subjects. This was done to test both on that we had captured the most common reading strategies, but also that the feature of "talking aloud" had not influenced the behaviour of the original subjects too much. In the second study we detected one new reading strategy, but else, the users in the second study used the same interaction strategies as those of the first. We can be fairly certain that the two studies together have captured some common behaviours of users in the domain, even if it is likely that a larger number of reading strategies would be detected. We here summarize briefly the results that are relevant for the task of filtering.

It turned out that it was very easy to distinguish between pure "browsing" and "search for particular information". Browsing occurred both at the group level and at the level of individual entries, and were in both cases implemented by the simple command "next". (This command was interpreted by the system as "next entry" while there were remaining entries in a meeting, or "first entry in next meeting" after the last entry of a meeting had been read.) Search, on the other hand, was signified by a user selecting certain entries or groups by clicking at them, or entering the command "subject next". Search on keywords in headers was less frequently used.

However, it turned out to be very difficult to recognize if the search succeeded, that is if the user really *found* interesting information. (Interest occurs of course on a floating scale, and none of our subjects ever found an entry that was directly relevant to him in his current work situation.) Originally, we thought that the reading strategy within a single entry could be used to signify interest. If a user would read for a long time, or go back and forth in an article, this would signify that the article was interesting for the user. However, we found several users who would exhibit this behaviour if *they weren't sure* if the article was interesting or not. This occurred usually when the header was vaguely interesting, but the content of the article was cryptic. In all observed cases, the users finally concluded that the article was not relevant to them.

We had better success in identifying reading strategies that signalled that an article or a topic was uninteresting. Articles that were read for a very short time were judged uninteresting "at a glance". Some subjects would select only such headers that were interesting for them, denoting that the ones they did not select were uninteresting for them. One of the subjects would also actively mark as "read" all messages in a topic that he found uninteresting. (This avoids getting them back in the next reading session as "unread".)

The most interesting feature of this domain however, is that users frequently would *change* their current strategy. Users frequently move from random scan to subject search (indicating an interest in a thread of articles with the same subject), or even directly from subject search to subject delete (indicating an uninteresting subject). This meta-level strategy of *pre-emption* of tasks seem to depend on the fact that users in this application in general estimate the likelihood of finding *any* interesting information to be small, and for that reason are prone to abandon strategies at an early stage, when the effort of searching seems unmotivated given the low likelihood of finding anything (in blunt words, when they grow bored).

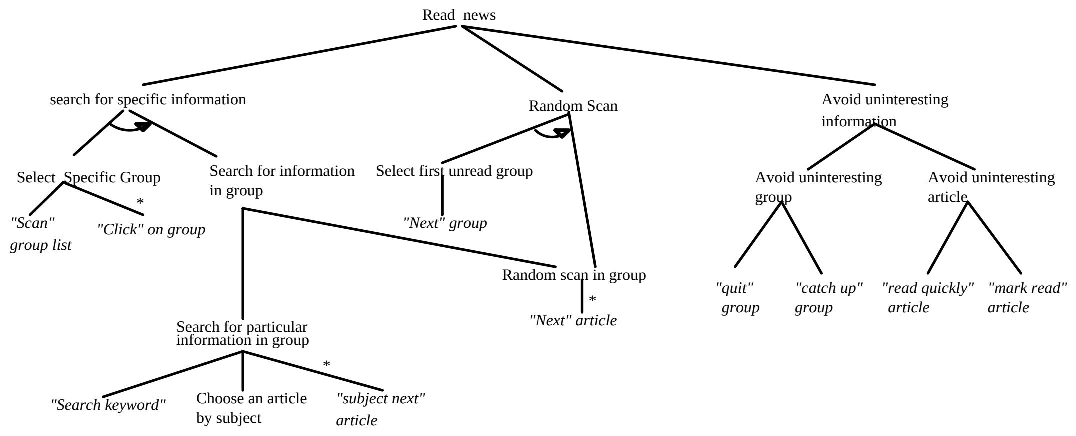
Read news
search for specific information
Random Scan
Avoid uninteresting information
Select Specific Group
Search for information in group
Select first unread group
Avoid uninteresting group
Avoid uninteresting article
"Scan" group list
"Click" on group
"Next" group
Random scan in group
"Next" article
"quit" group
"catch up" group
"read quickly" article
"mark read" article
Search for particular information in group
"Search keyword"
Choose an article by subject
"subject next" article

Figure 1: A task structure depicting the most common reading strategies.

3.2 Key actions and action sequence lengths

One interesting aspect we found was that we could identify "key actions" that would signify the user's current plan. Some actions could be a part of virtually any plan, whereas other were specific to one reading pattern. As an example, "read for a long time" could occur both during browsing and during search for some particular information, while "subject search" only was used when the users goal was to explore a certain topic.

Based on the knowledge about what we actually could detect, and the identified key actions, we have constructed a task hierarchy depicting the most frequent reading strategies for the news reading task, see figure 1. Most arches in the graph denote abstraction: by doing the low-level task, one achieves a higher level task. Arches that are connected with a curved arrow denote instead a sequential decomposition of a task into sub-tasks, and arches marked with a star denote an sub-task that may be repeated several times in order to achieve the higher level task. Observable actions in the interface are marked by italics.

Note that this task hierarchy is not the only one that could be constructed - this hierarchy is specifically formed to denote such tasks that *are possible to recognize* using keyhole plan recognition. Note also, that we cannot claim to capture *all* reading strategies - this was only a prestudy and the sample is very small. However, the task structure was formed after studying both novice and expert users of the tool and covers the strategies from both groups.

In total, we identified twelve key actions, and most tasks in the hierarchy can be identified after two actions have been observed. The only exception to this rule are the tasks "random scan" and "random scan in an interesting group", for which one must keep track of (in principle) an arbitrary number of actions, since we must remember *how* the group was selected throughout the reading actions within the group. However, these tasks are very closely related, and the fact that the group was once explicitly selected is not enough proof that the user has not moved over to "random scan". For the purposes of the application, the fact that a user selects

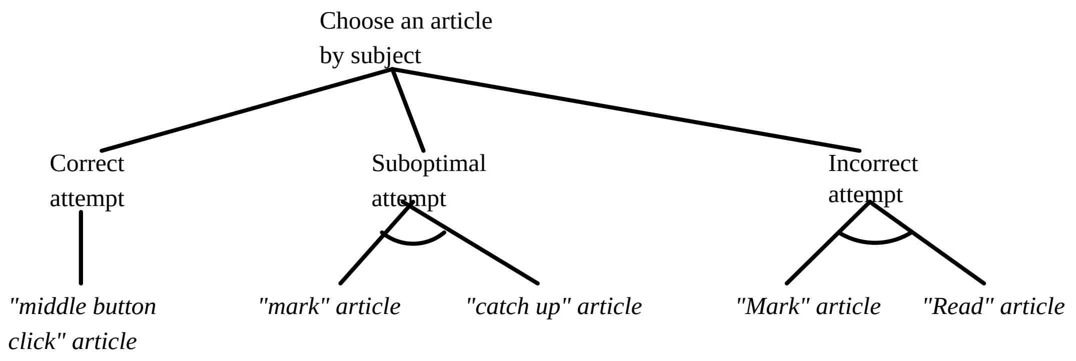

Figure 2: A correct, a suboptimal and an erroneous strategy for selecting an article.

a group is in itself enough sign of its significance.

3.3 Strategies of novices and experts

The empirical material shows some significant differences between novice and expert users, both at the level of individual interactions as well as concerning their overall strategies. Firstly, novice users were much more prone to stick to a random scan strategy. This could have been dependent on not knowing the tool, but the study indicated that it was due to users being unfamiliar with the information and its structure. Other differences in strategies were also recognised. In particular, experts were more prone to *delete uninteresting material* when found. The advantage of these strategies are that articles marked "read" will not be visible in the next session, and not accessible by "next" or "subject next". One expert utilized this strategy to the extreme - he started by *first* scanning for uninteresting material to delete, and only after that did he start to search for interesting material. Due to the differences in strategies between novices and experts, we believe that it would be possible to distinguish between different strategy preferences. This can be used to select appropriate help or tutoring strategies.

We also found some suboptimal and erroneous behaviours, see figure 2. An example of a suboptimal plan were that some users jumped to an article with a particular subject by marking it and then doing catch up, followed by read. This strategy achieves the task but leads to a side effect - all articles in between the sought article and the previously shown article are marked "read" , even though some of them might have been interesting for the user. There were also instances of erroneous actions for the same task - several users would mark the sought article header and then pressed "read". This action sequence does not achieve a jump - instead the next unread article will be displayed.

In our implementation, we have integrated both suboptimal plans and erroneous actions in the plan library. This falls into the "bug catalogue" approach to understanding user errors, which has been heavily criticised, since it is limited to recognizing such bugs that have been explicitly entered into the knowledge representation.

Calistri-Yeh [1991] has argued against this approach by claiming that

> "people are capable of an infinite variety of misconceptions, and it is quite unlikely that the plan knowledge base will have a pre-stored plan for the specific mistake that a user will make."

The main motivation for our choice is that the plan library should model the behaviour of the user group, as empirically established. From this aspect, empirically verified correct, incorrect and suboptimal strategies are just the same[1]. In our domain, we believe that most user errors can be captured by including a set of common erroneous strategies in the plan library. One should also note that in domains where novel strategies are frequent (whether they are erroneous or correct) keyhole plan recognition would be very hard or even impossible to achieve. The work by Calistri-Yeh deals with intended plan recognition, where the plan is explicitly stated.

4 Plan compilation and run-time "intention guessing"

From the previous section we can conclude that the information available for plan recognition in this task has an extremely simple structure. We have developed a plan compilation scheme aimed at dealing with this kind of simplistic plan recognition tasks. We have chosen to name this scheme "intention guessing" rather than plan recognition to distinguish it from more complex approaches.

From the task hierarchy, we can extract the behavioural patterns a user will exhibit while keeping to one task. This way, we describe a set of user *strategies*, each related to a specific task. The strategies are described using a set of simple, temporal relationships that we allow to be nested within each other:

- repeat A until A fails
- do A or B
- do A and then B
- do A and if A succeeds then B else C

Note that these structures contain only what can be externally observed, the actions and their effects. This way, they describe only the behavioural component of a user's plan, and omit the intentional aspects of it.

Based on the definition of a set of strategies, we construct a look-up table, in which each possible sequence of observable actions is related to the relative probabilities of the different strategies in which the action sequence occurs. This is done by viewing each strategy as a generating grammar for potential action sequences that fit this strategy. Note that some actions sometimes may *fail*, and that the action sequences for a strategy must contain both sequences for successful and failed attempts

[1]It is still useful to mark suboptimal or incorrect strategies as such, since this information can be used for tutoring purposes, but that use of keyhole plan recognition goes beyond the scope of this paper.

at executing the strategy. For example, if a strategy "(a then b)" is being executed, and a fails, it is not correct to continue with b. (In fact, most users will in this case repeat a, hoping that it may succeed the second time.) The construction of a set of action sequences for each strategy gives us a way of recognising failed attempts at tasks, and of distinguishing them from successful ones. This is inspired by similar work on integrating high-level planning with low-level execution mechanisms, see [Giunchiglia et al 1994].

The sequence generation process would not terminate by itself for strategies that contain "while" structures, and even if we restrict the size of execution sequences the number of sequences grows exponentially with the length of the included action sequences. For this reason, we restrict the intention guesser to monitor a "window" of the N latest observed actions. The size of the window can, if necessary, be slightly smaller than the average length of actions needed to uniquely identify the user's plan.

4.1 Run-time behaviour

At run-time, the intention guessing algorithm assigns a probability to each strategy, based on the currently observed window of actions. At any time, the current window will be almost identical to the window one action before, only shifted one step and one action added at the end. Thus, the possible windows that can be observed in a particular situation is heavily restricted from the previous step. We can utilise this fact and code the problem as a finite state automaton with A^{W-1} states and A edges from each state, where W is the length of the action window and A the number of recognisable actions. Such an automaton can be coded in Prolog utilizing indexing to give in principle constant time behaviour (if we ignore the time required to compute the index). However, that solution produces quite a lot of spatial overhead due to the construction of a very large number of predicates, as each node in the automaton must be coded as a separate predicate. In our implementation, we tried two other versions: one where the automaton was coded as a three-argument predicate, the first being the current node and the second the currently observed action, and one where the entire look-up table was kept. In the first, indexing was used to select the next state, whereas the second used sequential search. The advantage of the latter is that the table need only to contain entries for action windows that produce a non-zero probability for at least one strategy. Both implementations produced satisfactory behaviour on the readnews problem, but the latter was significantly less space-consuming.

Similar space savings could in some cases be obtained in the automaton approach. Since the automaton is constructed to accept any input string, what really signifies its behaviour is what *output* it produces. Thus, it can be viewed as a non-deterministic automaton for producing goal guesses rather than a deterministic automaton on input. Algorithms for transforming a non- deterministic automaton into a deterministic one can be found in standard literature on automata theory, and this can be utilized to reduce the automaton in cases where several inputs produce the same output.

4.2 Probability measures

We now turn to how to make a proper "guess", that is, how to calculate the probability that the user is executing a certain strategy given that a particular sequence of actions has been observed. This probability is dependent mainly on two factors: how many strategies the currently observed action window fits, and how large a portion of the window that fits the strategy.

Assume for now that all occurrences of the window in a strategy are full occurrences, that is, that the entire window occurs in the strategy. Let $N_{Window}(Strategy)$ denote the occurrences of a window in a strategy and M the total number of known strategies. Then

$$occurrence - weight(Strategy, Window) = \\ \frac{N(Window, Strategy)}{\sum_{i=1}^{M} N(Window, Strategy_i)}$$

There may also be occurrences when a part of a window fits a strategy, but not the entire window. This may indicate that a user has been executing the strategy, but stopped, or has just now started to execute a strategy.

Preemption of a strategy is handled in a very simple way. We simply required that the *last* action in a window must belong to the recognized strategy in order to constitute an occurrence. If a strategy partially fits the current window, but not the last action of the window, this signifies that the user may have *been* executing the strategy previously, but that this intention has been abandoned. Since we only are concerned with recognizing the current intention, we disregard this case entirely. In domains where users frequently interleave the executions of several plans, one may choose to count these occurrences as well, since this will cause all plans that are observed during the monitored action sequences to be recognized as alternatives.

Initialization of new intentions, on the other hand, requires careful treatment of the relative probabilities. Potentially, a user may have initiated a new strategy if the last action of the window belongs to the new strategy - but the probability of this change is lower than that of a strategy that fits the entire window. Furthermore, one must distinguish between initialization of a new strategy, where the partial occurrence in the window is an initial fragment of a strategy, and the resuming of a strategy, where the fragment may occur anywhere in the recognized strategy. In our domain, plan interleavings did not occur, why we again disregard the second case.

Still assuming that each strategy is equally likely to occur, initialization can be handled in the following way. Let$L(Window, Strategy)$ denote the average length of the portions of $Window$ that fit $Strategy$. Then the relative probability of a BP with respect to all other known strategies, given that the action Window has been observed is

$$partial - occurrence - weight(Strategy, Window) = \\ occurrence - weight(Strategy, Window) * \frac{L(Window, Strategy)}{\sum_{i=1}^{M} L(Window, Strategy_i)/M}$$

Finally, we may have obtained knowledge about the relative frequencies of different strategies. If we assume that the relative probability compared to all other known strategies is P(Strategy), we can calculate the relative probability of the strategy assuming that a certain window has been observed as

$$prob-weight(Strategy) \mid occurs(Window) = P(Strategy)*partial-occ-weight(Strategy, Window)$$

The sum of all probability weights is 1. A probability weight denotes only how likely a certain strategy is in relation to other known strategies. In order to know something about the true likelihood of a strategy, we must be able to estimate the likelihood that the user in the current situation is executing a strategy *outside* the known strategies. In our implementation, this is not done. The plan recognizer always returns one of two results: either a set of relative probability weights on a subset of strategies (always summing up to one), or else a zero probability for all strategies, signifying that the user's behaviour does not follow any known strategy.

To summarize, the implemented algorithm computes the relative probability of a strategy, given that a particular action window has been observed as

$$\begin{aligned} &probability - weight(Strategy) \mid occurs(Window) = \\ &P(Strategy) * \frac{N(Window,Strategy)*L(Window,Strategy)}{\sum_{i=1}^{M} N(Window,Strategy_i)*\sum_{i=1}^{M} L(Window,Strategy_i)/M} \end{aligned}$$

where $N(Window, Strategy)$ denotes the number of occurrences of $Window$ in $Strategy$, $L(Window, Strategy)$ denotes the average length of the portions of $Window$ that fit $Strategy$, $P(Strategy)$ denotes the probability of a user adopting a certain strategy, and M is the total number of known strategies.

5 Related approaches to plan recognition

The simplistic plan recognition mechanism has been specifically designed to deal with users *moving between tasks.* It has some important restrictions: it cannot recognise tasks at arbitrary levels, it does not handle plan interleaving, and it can only deal with strategies and strategy preferences that were foreseen at compile time. In this section, we compare our approach to existing approaches to keyhole plan recognition.

As noted in section 3.2, two of the higher-level tasks that are recognizable in our domain can only be distinguished from each other if an (in principle) infinite memory of actions was kept. Since the proposed intention guessing mechanism only keeps track of a limited number of the most recent actions, these tasks cannot be recognized using this technique. This situation is not a problem for the intention recognition mechanism in Kautz [1987]. This difference illustrates well the inherent "forgetfulness" difference of our approach. Kautz' approach, and any other that assume that users keep to one consistent plan as long as their actions are consistent with it, will have large problems with the frequent moves between different tasks that users exploit in this domain. This problem may be approached by *combining* the simplistic goal guessing mechanism with a more advanced approach, using the simplistic intention guesser for recognizing low-level actions, and a traditional plan recognition mechanism for recognizing high-level plans. This combination becomes particularly interesting in interfaces that combine task-oriented dialogue with direct manipulation - we present an application example of this in the next section.

Similarly to Kautz, Quast [1993] proposes a deductive approach to plan recognition, but where arbitrary interleavings of plans are allowed. The forgetfulness of our intention guesser does not allow us to recognize that a previous strategy is resumed. However, Quast cannot handle *preemption* of plans - the only way to deduce that a plan no more is intended is if it is completed. It is not even clear if Quast's approach can handle the case when a user abandons a plan because it is *impossible to achieve* - this is in our approach handled to some extent by recognizing failed attempts at strategies. In the domain we studied, *no* interleaving of plans occurred, whereas plan pre-emption was frequent. Our current experience does not allow use to draw any conclusions about which is the more frequent behaviour, and there are most likely applications of both kinds.

We believe that a correct treatment of plan interleaving and plan pre-emption in deductive approaches would require that the plan library contained *explicit knowledge* about how users move between tasks (c.f. [Wilensky 1981] on meta-knowledge about the planning process) - in particular since this property is application dependent. Viewed in this light, the forgetfulness of our algorithm provides an extremely cost-effective shortcut for dealing with domains where plan pre-emption is frequent.

The intention guessing algorithm is completely dependent on a precoded plan library. This problem is well-known in literature on plan recognition, and it is unlikely to ever get a satisfactory solution for keyhole plan recognition. There exists solutions for intended plan recognition that can deal with recognizing instances of novel plans or user-specific preferences or misconceptions [Pollack 1986], [Calistri-Yeh 1991]. However, the only possible approach to achieving any of this in keyhole plan recognition is to introduce some mechanism of *learning* over time based on probabilistic data, and it is unclear if such training can be achieved that can recognize completely novel plans. Bauer [1994] presents an attractive mechanism for adapting a plan recogniser to the particular interaction patterns of individual users. After a number of training sessions, the system is able to detect the next most likely action for a user from an observation of one (or several) immediately preceding actions.

The training mechanism that Bauer proposes does not associate any *user intention* to the different behavioural patterns. Bauer assumes that this information is provided by some external source, either from a precoded plan library or supplied by a user-specific planner (as is described in [Bauer et al. 1991]). In our framework, we extract the strategies from a task structure. This way, we achieve that each strategy has a task associated to it, giving an intentional interpretation of the user's behaviour. It is possible that the work in [Bauer 1994] could be merged with our approach, yielding a way to train the precoded mechanism to fit individual users.

A mechanism that is specifically geared to training a system to recognize user behaviour has been proposed by Ingrid Zukerman (personal communication). This mechanism is directed to *obstructed* plan recognition in game playing, in this case the game of MUD. This system supposes that information can be obtained that attach an overall goal (a "quest") to an observed sequence of user actions. So far, the system has been trained on a set of prepared examples, containing both an observed action sequence, and the quest it is intended to achieve. Again the problem is if and how the information about "quests" associated to behavioural patterns could be observed in other situations than from training examples, but due to the nature of the domain it is not altogether inconceivable that a quest can be recognized *once it has been fulfilled.*

The trained mechanism proposed by Zukerman recognizes plans in an even more simplistic way than our compiled intention guesser. Her behavioural patterns in describe only the relative frequencies of different actions in all plans for a certain quest. The system must keep track of the relative frequencies of all actions since the quest started, and it cannot form strategies for disregarding certain actions that occur in strategies for a large number of quests, or recognizing actions at the right level of abstraction. Finally, it is not clear how this mechanism can be adopted to cope with preemptions of quests and initialization of new quests. In all these respects, the intention guesser has advantages. Note however that a precompilation strategy is in general too weak to deal with obstructed plan recognition, since as soon as the user catches on to the plan recognition mechanism, he or she will modify his/her behaviour to fool the plan recognition mechanism.

6 Integrating intention guessing with intended plan recognition

The intention guessing mechanism has here been proposed as a mechanism for simple and efficient keyhole plan recognition in simple domains, but it has an entirely different usage as well, in interacting with a more complex system for intended plan recognition. This would allow integrated plan recognition for a system that contains both a task-oriented dialogue, as well as direct manipulation components. In such a system, we can expect that users will exhibit a high-level task-oriented behaviour while still moving frequently between different low-level tasks, or strategies for the same low-level task.

We are currently exploring this idea in a separate project aimed at developing a software assistant system. The system presents documentation of a software development method. Currently, this project has developed a system in which information may be found both by point-and-click navigation through the information structure, and also by posing direct questions from a limited query menu [Lemaire et al. 1994]. Questions can concern individual objects, but also be comparisons between two objects, or more generally, about two concepts. Comparison questions can be posed as "follow-up questions" to other questions. In this application, we have found plan recognition a useful means to select *answers to questions*, while the navigation through the information structure still is done by direct manipulation. (We are also including some other means of adaptation, see [Höök et al. 1994].)

The application integrates a (partly) task-oriented dialogue with direct manipulation. The user's navigational manoeuvres can be viewed as "keyhole" sources of information, in a manner similar to that of the news reader interface. However, once users indulge in a question - answer dialogue, we will also give them the means to *explicitly* state their intentions.

7 Conclusions

When applying plan recognition to Human - Computer Interaction, one must cope with users exhibiting a large amount of reactive behaviour without notifying the system: users that change tasks, or change strategies for achieving tasks. To support this claim, we have presented an empirical study of an application of plan recognition where users exhibit reactive rather than plan-based behaviour, and where existing approaches to plan recognition do not perform well. We have described an extremely simplistic mechanism for keyhole plan recognition, "intention guessing", to deal with this type of domains.

The simplistic "intention guessing" algorithm is based on descriptions of *observable behaviour*, and is able to recognize certain instances of plan failures, suboptimal plans and erroneous actions. At run-time, the algorithm only keeps track of a limited number of most recent actions, which makes the algorithm "forgetful", which is an extremely useful property in domains where users frequently abandon strategies. A useful extension of the "intention guessing" mechanism is to integrate it with a high-level mechanism for intended plan recognition.

The simplicity of the intention guessing algorithm leads to a number of restrictions:

- The algorithm cannot always distinguish between high-level strategies where long sequences of actions are the same,
- it cannot handle arbitrary interleavings of plans, and
- it is dependent on a full compile-time characterization of the plan recognition task.

On the other hand, it has the following advantages over other approaches to keyhole plan recognition:

- the very efficient runtime behaviour,
- it copes well with preemption and initialization of plans

It is an open question to what extent users exhibit plan interleaving and plan preemption in HCI applications. It is likely that this property is highly domain-dependent. Furthermore, in particular HCI applications it may prove very difficult to recognize high-level user strategies, due to the low level of individual interactions with the system. For these reasons, one must carefully analyse any plan recognition application to deduce *what* goals are recognizable, and how persistent users are in pursuing these goals, before devising a plan recognizer that is suitable for the particulars of the applications.

8 References

M. Bauer, S. Biundo, D. Dengler, M. Hecking, J. Köhler, and G. Merziger. 1991. Integrated plan generation and recognition: a logic- based approach. In Proceedings of the 4th Internationaler GI- kongress Wissenbasierte Systeme, München, pages 266 - 277, Springer IFB 291.

M. Bauer. Quantitative modeling of user preferences for plan generation. 1994. In Proc. of the 4th international conference on User Modelling, Hyannis, Mass., available from MITRE Corp. pages 73- 78.

R. J. Calistri-Yeh. Utilizing user models to handle ambiguity and misconceptions in robust plan recognition. 1991. in *User Modelling and User-Adapted Interaction 1*, pages 289-322.

F. Giunchiglia, L. Spalazzi and P. Traverso. Planning with failure. 1994. in Proceedings of the 2nd International Conference on AI Planning Systems (AIPS-94). Chicago, Illinoi. AAAI press.

K. Höök, J. Karlgren and A. Wærn. 1994. A glass box intelligent help interface. Submitted to CHI'95, available from SICS.

H. A. Kautz. A formal theory of plan recognition. 1987. Ph.D. thesis, technical report 215, dept. of Computer Science, University of Rochester.

J. Karlgren, K. Höök, A. Lanz, J. Palme and D. Pargman. 1994. The glass box user model for filtering. In Proc. of the 4th international conference on User Modelling, Hyannis, Mass., available from MITRE Corp.

R. Kozierok and P. Maes. A learning interface agent for scheduling meetings. 1993. in proc. of the int'l workshop on intelligent user interfaces, Orlando, Florida. ACM Press.

B. Lemaire, C. McDermid and A. Wærn. Adaptive Help by Navigation and Explanation. 1994. SICS technical report T94:05.

D. J. Litman and J. F. Allen. A plan recognition model for subdialogues in conversations. 1987. *Cognitive Science 11*, pages163-200.

M. E. Pollack. A model of plan inference that distinguishes between the beliefs of actors and observers. 1986. in Proc. of the 24th annual meeting of the assoc. for Computational Linguistics.

K. J. Quast. Plan recognition for context sensitive help. 1993. in Proc. of the 1993 int'l workshop on intelligent user interfaces, Orlando, Florida. ACM Press.

L. Suchman. *Plans and situated actions. The problem of human machine communication.* 1987. Cambridge university press.

R. Wilensky. Meta-Planning: representing and using knowledge about planning in problem solving and natural language understanding. 1981. *Cognitive science 5*, pages 197 - 233.