Abstract—Interface agents are strategic software components for improving the quality of services to users. To be accepted by users, interface agents need to make useful suggestions always in the context of the user’s intention. The user’s intention should be detected as soon as possible so that the agent can define a way to collaborate with the user. Plan recognition can be applied to identify the user’s goal based on his or her actions in the environment. However, classical approaches to plan recognition fail in two main aspects that make them unsuitable for being used by interface agents: the lack of personalization and the lack of consideration of the transition between different goals pursued by the user. We propose an approach to capture intentions taking into account the variables involved in the application domain that represent the user preferences. Experimental evaluations show us that we have found a way for early detection of intentions.

Keywords—Plan Recognition, Interface Agents, User/Machine Systems, Knowledge personalization and customization.

1 INTRODUCTION

Interface agents [1] are computer programs designed to assist human users in their computer-based tasks in a personalized manner. This kind of agents is able to learn the interests, preferences, priorities, goals and needs of the user aiming at providing him or her proactive and reactive assistance in order to increase the user's productivity regarding to the use of the application at issue. With the aim of assisting a user of a software application, interface agents not only have to learn the user's preferences and habits regarding the use of the application itself, but should also consider the intention he or she is involved in before initiating an interaction with the user. Considering the status of a user's attention (his or her intention or the goal he or she is pursuing) and the uncertainty about the user's intentions are critical factors for the effective integration of automated services with direct manipulation interfaces [2]. As a result, we must build agents capable of detecting the user’s intention so that they can predict opportune moments for gaining the user's attention.

Most of us have experience in some moment the situation in which a software application foretold what we were intending to do. A typical example of such application is the Office ‘97’s
assistant, Clippy, who interrupts us with phrases like “It looks like you are writing a letter…”

However, Clippy tells us the same phrase whenever we start writing a sentence beginning with the word “Dear …”. So, in these kinds of interventions Clippy is not capturing well the user’s intention.

A correct detection of the user’s intention will be useful for the agent to avoid interrupting the user in an improper moment. Users generally don’t want to be interrupted while working on a specific task, unless this interruption is strongly related to the task they are performing. For this reason, most users of Clippy find it annoying and wind up not using it [3]. By considering the user's intention the agent will be able to answer to the user’s requirements always in the realm of his or her current goal. For example, if the agent observes that the user is scheduling a work meeting for the following day, the agent can offer to automatically complete the information required and to send an email to each participant of the meeting, providing that it knows the user's preferences about the kind of meeting he is scheduling.

With this purpose, plan recognition aims at identifying the goal of a subject based on his or her actions in the environment. Most of previous approaches to the problem of plan recognition fail in two main aspects for being used by interface agents. First, one of the most important problems that an interface agent faces when inferring the user's intention is the uncertainty related to the moment in which the user starts a new plan to achieve a new goal, that is how does the agent become aware that the user have already achieved one goal and started pursuing new one? This issue is not usually addressed by many approaches to the problem of plan recognition, and they consider only one "session" which starts with the first observed action and ends when the algorithm recognizes the user's intention. In an interface agent environment, the user will repeatedly start pursuing new goals in the application, with no preplanned behavior. Moreover, the user can even change his or her intention without completing his or her previous goal. This
problem is usually tackled by restricting the memory of the plan recognizer so that it only considers the most recent tasks performed by user, or it considers each task for only a fixed interval of time and then they are completely disregarded. We present a novel alternative approach in which the strength of the evidence of the user’s actions is decayed over time.

Second, most of previous approaches do not consider adaptation to a particular user. Plan libraries are only domain dependant, and not both domain and user dependant, because they are constructed beforehand according to dependencies in the tasks of the domain and not considering that the behavior may vary from one user to another according to the values taken by some variables of the domain. Some variables define different context in which the user may a different behavior. For example, receiving an invitation for a meeting gives one context for the user’s subsequent behavior, but if it comes from his or her boss, the user may behave completely different.

To face these problems, we propose an approach that will allow an interface agent to perform personalized plan recognition over the tasks the user executes in a software application. We will not focus on how the agent makes use of the information about user intentions to provide assistance but on providing the agent with a probabilistically ranked list of the most probable intentions the user may have in every moment. The main contributions of our approach are the personalization of the plan recognition process and the use of a fading function for evidence to face the problem of transitions between goals pursued by the user. This paper is structured as follows. In Section 2 we present the problem of plan recognition. In Section 3 and 4 we present and evaluate our approach. Finally in Section 5 we present our conclusions.

2 WHAT IS PLAN RECOGNITION?

Consider the following situation. Mary, a secretary, observes her boss opening a personal
information management software application that is well known by both of them. This software application has many tools including managing events, contacts, reminders, home budget, personal diary, lists of favorite web pages, passwords, etc. He can use any of the tools the software application offers. Next, she observes her boss clicking on the "Add event" button. Now, she is sure he wants to add a new event to the calendar. She knows the steps required for completing that task, and she is able to help him in any moment. The boss enters the subject of the event, the day and starting time and then presses the "Add participants" button. After scrolling the list of contacts, he presses the "Cancel" button and saves the event. She cannot imagine what he is intending to do. Perhaps he did not find the contact he was looking for, or maybe he has a meeting and cannot spend more time scheduling the event. But next the secretary observes the boss pressing the "Add contact" button, completing the required information and saving it. With the information the secretary has so far, she can think that the participant the boss was intending to add was not in his or her contact list so he had to add it, and after that he will edit the event he had saved before and add the new contact to it as a participant. Moreover, if the secretary has enough knowledge about his or her habits and the events he usually adds to the application, she could be able to predict, for example, that the event will take place in a specific location, at a specific time just by observing the subject, the host and other information he has entered. It may also be possible for her to predict that some member of the office will take part of the meeting based on the type of meeting the boss is scheduling.

The term that has been introduced to describe the process of inferring the intentions of a subject based on the actions he performs in an environment is plan recognition. What the secretary in the previous example has done by trying to deduce her boss intention can be considered a plan recognition process. Inputs to a plan recognizer are generally a set of goals the agent expects the user to carry out in the domain, a set of plans describing the way in which the
user can reach each goal, and an action observed by the agent. The plan recognition process itself, consists in foretelling the user's goal, and determining how the observed action contributes to reach this goal. The set of possible plans to achieve goals makes up a plan library.

There are currently two main approaches to the problem of plan recognition: consistency and probabilistic. Consistency approaches [4], [5], [6], [7] aim at narrowing the set of candidate intentions by eliminating those plans that cannot be explained by the actions that the user performs. Probabilistic approaches [2], [8], [9], [10], [11], on the other hand, explicitly represent the uncertainty associated to the user’s plans and allow a probabilistic ranking of the user's intentions.

Both kinds of approaches can lead to accurate predictions providing that the plan library is complete in consistency approaches, and that the probabilities are correct in probabilistic approaches. However, probabilistic approaches can find the most probable intention if the observations up to a given moment enable more than one possible intention, while consistency approaches cannot select between them and has to wait for a single consistent explanation. See [12] for a review of plan recognition for interface agents.

There are two main aspects that make classical approaches to plan recognition unsuitable for being used by interface agents. First, the agent should deal with transitions and changes in the user intentions. The agent will usually not be certain about the moment in which the user starts a new plan to achieve a new goal. Classical approaches do not consider this problem and they restrict the plan recognition process to only one "session" that starts with the first observed action and ends when the algorithm recognizes the user's intention. In an interface agent environment the user not only will repeatedly start pursuing new goals in the application, with no preplanned behavior, but also he may change his or her intention without completing the previous one. Those approaches which consider this problem restrict the memory of the plan
recognizer so that it only considers the most recent tasks performed by user, or it considers each task for only a fixed interval of time and then they are completely disregarded.

Second, user preferences have a fundamental role for an interface agent and should be considered in the plan recognition process. Considering the user’s preferences in the plan recognition process is an important factor that an interface agent should regard because the behavior of the user in a specific situation is usually determined by his or her preferences related to that situation. In this context, a situation is described by the information the user is handling to achieve his or her intention. In the calendar scheduling domain, for example, the situation related to a new event addition may be described by its date, time, participants, place, subject, etc. These variables of the domain strongly influence the user behavior. For instance, if the user adds a new event which time is in his or her working hours, there is a high probability of the event taking place in the office.

The process of adapting a computer application to the needs of a specific user is called personalization and takes advantage of knowledge acquired from the analysis of the user's behavior and data. In contrast to customization which is a user-initiated and user-driven process, personalization is system-initiated and system-driven and thus requires the system to monitor the user's behavior in order to adapt automatically. The fundamental purpose of personalization is the user's satisfaction, and it is motivated by the acknowledgment that a user has needs and preferences, and meeting them successfully is likely to lead to a satisfying relationship with him.

The importance of personalizing the plan recognition process by considering user preferences has been identified previously in [14] where decision trees are used to learn classes of situations based on user actions. This approach runs the plan recognizer on a set of input examples from a typical user, and then gathers statistical data based on the results of running the plan recognizer on the entire observable behavior in each episode. The main problem of using decision trees to
"classify" the intention of the user according to some attribute of his or her actions is that they can only ascribe one possible intention given those attributes, completely discarding other options.

In the next Section we present our approach to provide an interface agent with a probabilistic ranking of user intentions which take into account the user’s preferences.

3 Probabilistic Ranking of User Intentions

In general, a user performs a set of tasks in order to achieve a higher level goal. For example, to arrange a dinner with his or her friends, the user has to look at the calendar for a convenient date, he or she has to add a new event to his or her agenda and write an invitation mail to each participant of the meeting. Therefore, when the user performs a task, the agent should consider the set of possible intentions associated to it to find a way to collaborate with the user. The prediction of the user's intention is an inherently uncertain task. Therefore, we look for a knowledge representation capable of capturing and modeling dynamically the uncertainty of human-computer interactions.

In our approach, we represent the set of intentions the user can pursue in the application domain as an Intention Graph. We define an Intention Graph to be a representation of the context of execution of tasks and it is materialized by a Bayesian network. The context is viewed as the set of tasks that the user has performed recently, and will influence the confidence that the agent has in any given intention that the user may be pursuing.

Bayesian Networks are a probabilistic knowledge representation used to represent uncertain information [12]. Bayesian networks are directed acyclic graphs representing both the conditional dependencies and independencies between elements of the problem domain. The knowledge is represented by nodes called random variables and arcs representing the causal
relationships between variables. Each variable has a finite set of mutually exclusive states. Nodes without a parent node have an associated prior probability table. On the other hand, the strengths of the relationships are described using parameters encoded in conditional probability tables (CPTs).

Bayesian networks are used for calculating new probabilities when some particular information becomes available. The information is called evidence and will be in the form "A=a", where A is a variable and a is a state of A. In our case, we will have new evidence every time the user performs a task in the software application. Therefore, evidence will be in the form “AddContact=true” meaning that the user performed “AddContact” task.

The general setting for probabilistic inference is that we have a set \( V = V_1, V_2, ..., V_k \) of propositional variables and we are given as evidence that the variables in a subset \( E \) of \( V \) have certain definite values, \( E=e \). In the probabilistic context, we want to calculate the conditional probability distribution \( p(V_i|E) \), for the variable \( V_i \) given the evidence. This process is called probabilistic inference.

By making use of probabilistic inference, we will be able to know, given as evidence the set of tasks performed by the user, the probability that the user is pursuing any given intention modeled by the Intention Graph. Moreover, if the user explicitly declares his or her intentions, we will be able to probabilistically assess the tasks he has to perform to achieve his or her goal.

### 3.1 Building an Intention Graph

In our Intention Graph variables correspond to goals that the user can pursue in the application domain and to tasks that the user can perform in the application to achieve those goals. The two possible states of these variables are \textit{true} and \textit{false}. A \textit{true} value in a variable representing a goal indicates that the user is pursuing that goal. On the other hand, a \textit{true} value in a variable
representing a task indicates that the user performed that task in the application. We call confidence level to the probability of a variable being in state \textit{true}. Notice that we will not have direct evidence about the goal the user is pursuing unless the user makes an explicit declaration of his or her intention. Evidence on a task node will be set when the user interacts with a widget in the application GUI that is associated to the execution of that task. Our Intention Graph included a third kind of variable: context variables. This kind of variables will be use to personalize the intention detection process and will be explained later in Section 3.4.2

In order to construct the graph, an expert in the domain of the application should first add a node for each variable in the domain and then draw arcs according to existing relationships among variables in the domain. Missing arcs encode probability conditional independence assertions, as no dependency exists among unrelated nodes. Considering that the tasks the user performs are caused by what he intends to do in the application environment the direction of arcs in our Intention Graph is always from goals to tasks.

Once the structure of the Intention Graph is built, the domain expert should assess the strength of the relations between variables my means of conditional probabilities tables in each variable with parents and a priori probabilities for variables without parents. Although the probabilities selected will have a strong influence in the quality of the Intention Graph, they can be automatically updated later by using the user feedback about his or her intentions, as we will detail in Section 3.4.

\textbf{3.2 Example}

To illustrate our proposed approach we will use an application called \textit{Agenda 4.0} that provides various functions for managing personal information such as contacts, tasks, events, and email.

The calendar domain is a well known domain in which interface agents can be applied. An
interface agent in this domain can assist users with tasks such as suggesting meeting places, dates and times for a new event the user is scheduling, suggesting the participants of a meeting, notifying the user when an event is about to happen and warning about overlapping events and about an unusual event scheduled by the user. As stated before, to provide this functionality, the agent should be aware of the task the user is performing because it gives the context in which the user is moving through. This context will allow the agent to infer the intention of the user, so that it can provide assistance to him or her if needed, or it can collaborate with the user.

The Intention Graph we designed for this application is shown in Fig.1. Nodes labeled with black background and white text correspond to intention nodes and nodes labeled with white background and black text correspond to task nodes.

As we can see, we identified six intention nodes: "InviteToMeeting", "SendMailToContact", "RegisterBirthday", "ContactBirthday", "EventBirthday" and "AddMissingContactGroup". "InviteToMeeting" reflects the intention of the user of arranging a meeting with a selected contact. On the other hand, "SendMailToContact" reflects the intention of the user of sending an email to the contact he has selected in the application. "RegisterBirthday" is a goal that can be achieve either by registering the birthday of a selected contact in the personal information of that contact ("ContactBirthday" intention) or by creating a new event to remember about that important date ("EventBirthday"). Finally, "AddMissingContactGroup" is used to detect when the user has to add a new contact group to the agenda to be used afterwards as the group of a given contact. Although a simple example, in Section 4 we show how beliefs on the intentions modeled in the Intention Graph are updated as a user interacts with the application.

### 3.3 Fading Past Observations

Recall the example of the secretary presented in Section 2. What should she think if before
completing the intention the secretary believes her boss has, he receives a telephone call and she observes that he makes a new entry in his personal diary in the application? Should she consider that her belief was wrong? He can yet continue editing the event after saving the new entry in the diary. Or perhaps her boss abandoned his previous goal and started a brand new one. Therefore any plan recognizer must be careful not to over-commit themselves to one possible intention. Especially in a domain where users can change they current goal rapidly, it is often a good idea for the plan recognition system to be a little "forgetful".

As stated in Section 2, most of previous approaches to the problem of plan recognition do not consider the uncertainty related to the moment in which the user starts a new plan to achieve a new goal. Those which consider this issue limit the memory of the plan recognizer by making evidence to be present in a fixed interval of time and then completely disregarding it. We take a different approach in which evidence is gradually forgotten.

In Bayesian networks terms, evidence is a collection of findings on variables. A finding may be hard or soft. A hard finding specifies which value a variable is in. Findings on the values taken by the variables introduced before are of this kind. A soft finding, on the other hand, specifies the probability distribution of a variable. Hard evidence is a collection of hard findings and Soft evidence is a collection of soft findings.

In our approach, we adopt the concept of soft evidence to fade the evidence we entered to the Bayesian network as the user performs further tasks. To do so, we use a fading function $F(\cdot)$ to gradually forget the tasks performed by the user, as shown in Eq. 1.

$$P(t = true) = P_{prev}(t = true) - F(\cdot)$$  

Eq. (1)

That is, the current probability that task $t$ was performed $P(t=true)$ is computed by subtracting from the previous probability assigned to task $t$ $P_{prev}(t=true)$ a value assigned by the fading
function $F(\cdot)$. Evidence on every node is faded according to this function until they reach their original value, that is until the probability of a given node becomes less than the value that it would have if we wouldn't have observed the execution of the corresponding task in the application.

By gradually forgetting past observations, the agent not only would be able to manage changes in the user’s goal, but also will allow it to forget the execution of noisy tasks, that are tasks that do not belong to the main goal the user is pursuing. In our example the boss can check the current date for its own sake, and not because it is part of the plan he is pursuing in that moment.

Fading functions can be any function that, given the Intention Graph and the evidence on tasks performed so far, decrements the certainty of the evidence gradually according to some heuristic [14]. For example, we can decrement current evidence by a fixed factor $0 \leq \Delta \leq 1$ every time the user performs a task in the application. This way, for all evidenced nodes $t_i$ we will update the probability distribution of its evidence according to Eq. (2)

$$P(t_i = \text{true}) = P_{\text{prev}}(t_i = \text{true}) - \Delta$$

Eq. (2)

This simple fading function will allow the agent to disregard, after some actions performed by the user, a previously performed noisy action or a previously pursued goal. This function, however, might have the problem of rapidly forgetting evidence in tasks that actually contribute to the current goal of the user. This problem can be overcome using a more sophisticated function that, for example, keeps almost intact the evidence of some number of tasks and then quickly decrement up to the original value of the node (without evidence).

### 3.4 Personalizing the Detection of User Intentions

When a plan recognizer is to be adapted to a particular user of a software application, a proper model of his or her preferences is crucial because preferences are the reasons why a user acts in a
particular way when facing a certain situation [15]. In our example, the secretary should learn that her boss prefers organizing events in his house when the participants are his friends or when it takes place in the afternoon and in the club when it takes place in the morning. Not knowing about the user's preferences hinders the plan recognition process, especially if the behavior of the user deviates from what is typically to be expected in a domain. So, an agent provided with plan recognition and user modeling capabilities can detect patterns of behavior that are particular to each different user of the application.

In our approach, we personalize the Intention Graph initially constructed by the domain expert in two ways, by adapting the probabilities (Section 3.4.1) and by learning new relations (Section 3.4.2).

### 3.4.1 Adaptation of Probabilities in the Intention Graph

As stated before, Bayesian networks inference mechanism will allow the agent to update its belief in any given goal related to the tasks performed by the user. This belief update is made using the probabilities set by the domain expert who designed the Intention Graph. However, when constructing a Bayesian network, the domain expert may be uncertain of the correctness of the conditional probabilities chosen. However, when the agent is at work, it will repeatedly get new experiences of tasks performed by the user pursuing some particular goal. It would be desirable that the agent could learn from these new cases. To adapt the probabilities set by the domain expert who constructed the network to the way of acting of a particular user, we take a statistical on-line learning approach.

As the user interacts with the application, the agent “remembers” the tasks performed by the user. When the agent infers that the user is pursuing certain goal, it adapts the probabilities of the intention graph to reflect this new experience. The agent will believe that the user has already completed the current goal(s) when a certain threshold is exceeded. This threshold should be
empirically determined for each application domain. Explicit feedback can also be considered if
the agent believes that it can ask the user whether he is trying to achieve a given goal or not, but
this decision is out of the scope of our work.

The statistical method we used to update the CPTs of the variables in the network is called
Fractional Updating and was proposed by Jensen [16]. The statistical task is to gradually modify
the estimates of the parameters with the new experiences entered. To do so, besides the CPT for
each node, we keep an experience table. The experience table of a variable represents the
experience counts of the parent configurations. For example, in the Intention Graph in Fig. 1, the
experience table for the node labeled “Events” will have two entries, one indicating the number
of times we observed the execution of “Events” task when the user intended to register an event
to remember about a birthday (“Event-Birthday” intention) and the other indicating the number
of times we observed the execution of “Events” task when the user was not pursuing
“Event-Birthday” intention.

Let $n_k = P(N=k)$, the probability of node $N$ being in state $k$, $y_k = P(N=k|e)$ the probability of node
$N$ being in state $k$ given the current evidence in the Bayesian network, $conf$ a given configuration
of the parent nodes of $N$, $z$ the probability of such configuration given the current evidence and
$\text{experiences}\_\text{prev}$ the previous experiences count for $n_k$. We update the CPT for node $N$ being in its
state $k$ for a given parent configuration $conf$, according to Eq. 3.

$$P(n = k | conf) = \frac{n_k + z \cdot y_k}{\text{experiences} \_\text{prev} + z} \tag{3}$$

For further information about Fractional Updating, please refer to [16].

3.4.2 Learning New Relations in the Intention Graph

A further adaptation of the Intention Graph built by a domain expert can be done by learning
new relations that may arise between the attributes of the tasks performed by the user and the
intention nodes in the Intention Graph.

For example, in Agenda 4.0 application, the user can select a contact from the address book with the objective of sending this contact a mail or with the objective of scheduling a meeting with this contact, as is shown in the Intention Graph presented in Fig.1. The Intention Graph constructed manually by a domain expert will allow the agent to rank which of these two goals is more probable, given that the user selected a contact in his or her address book. However, the information of the selected contact can be relevant in discerning which goal the user is pursuing.

To consider this information, we introduce to the definition of our Intention Graph, the concept of traceable nodes. A traceable node is a node of the Intention Graph in which we want to register the values taken by some attributes of the corresponding task performed by the user with the aim of adding new variables that represent the context in which the user performs that task and to find new relations between these variables and the nodes in the Intention Graph.

In the example above the task corresponding to the selection of a contact in the address book is a traceable node. The designer of the Intention Graph should decide which attributes of this task are interesting (for example, the city in which the contact lives or the group the contact belongs to) for which set of intentions (in the example, sending a mail to the selected contact or scheduling a meeting with him or her).

Every time the user performs a task corresponding to a traceable node, the agent will observe the values taken by the attributes of the task (for example, the selected contact is from New York and belongs to the group of friends). Then, the agent will continue observing the user until it can infer which his or her intention(s) are and will save the experience in an interaction history. Each experience will be in the form

\[
<\text{attribute}_1, \text{attribute}_2, ..., \text{attribute}_n, \text{intention}_1, ..., \text{intention}_k>
\]

where \text{attribute}_i is the value taken by the attribute \text{i} and \text{intention}_j is \text{true} if the agent infers that
the user was pursuing intention \( j \) or \( false \) otherwise.

This database of experiences is then used by the agent to run both a batch learning and a parametric learning algorithm to find relations between the attributes themselves and between the attributes and the intentions. Fig. 2 shows an example of a learnt network for the example we are following.

In this machine-learned Bayesian network, variables representing attributes are incorporated to the network as a new kind of variables, context nodes (gray nodes correspond to this kind of nodes), and variables representing intentions as intention nodes. The learnt network is then merged with the Intention Graph to incorporate this knowledge to the detection of the intention of the user.

4 Experimental Results

To perform our experiments, we selected the calendar management domain for two reasons: first, because it is a well known and easy to communicate domain in which interface agents can be applied and second because of the richness in the number of variables involved which make it a non-trivial domain. An interface agent in this domain can assist users with tasks such as suggesting meeting places, dates and times for a new event the user is scheduling, suggesting the participants of a meeting, notifying the user when an event is about to happen and warning about overlapping events and about an unusual event scheduled by the user. A complete development of such an agent is out of the scope of this work. We are concerned just in providing an ordered ranking of the possible goals the user can pursue.

With the objective of showing the influence that the user's preferences have in the detection of the user's intention, we select a scenario observed from the interaction of a regular user of the scheduling application in which he used the application to organize a meeting with some contact
in his address book and then selected another contact to register his or her or her birthday. To achieve these goals, the user performed the following sequence of tasks: <"SelectContact", "AddContactToMeeting", "SelectContact", "EditContact", "PersonalInformation", "EnterBirthday">

We recorded the confidence levels for all intentions both in the Intention Graph containing the user’s preferences information and in the same Intention Graph without context nodes (the one owned originally by the agent). Figure 3.a shows the evolution of the confidence levels of each possible intention in the original Intention Graph without context nodes. In the first time slice, we show the a priori probabilities of each intention when the user did not perform any tasks in the application. "SendMailToContact" is the more probable intention, while "ContactBirthday" is the least probable one. When the user performed the first task, "SelectContact", the ranking remained unchanged, although there was a little increment in those intentions that contained this task. Then the user performed "AddContactToMeeting" and "InviteContactToMeeting" became the most probable intention. With the following set of tasks performed by the user, "ContactBirthday", "RegisterBirthday" and "EventBirthday" started gaining confidence while the other intentions decreased its confidence level. "AddMissingContactGroup" confidence level remained unchanged along this session because it is "disconnected" of the tasks performed by the user. The agent in this case considered a threshold level with a value of 0.7 to believe in the intention pursued by the user; it predicted the first intention with the second performed task and the second intention in the fifth task. If we consider the number of tasks in each intention, it needed both tasks to be performed to detect "InviteToMeeting" intention, and three tasks out of four to detect "ContactBirthday" intention.

Fig.3.b shows the same scenario but performed using the Intention Graph with context nodes merged. The first "SelectContactTask" was performed when the user selected a contact from the
"Friends" group, living in "New York", and that already has the birthday registered. We can see that the confidence level for "InviteToMeeting" is higher only with the first task performed. We can also see that the other intentions dramatically lowered their confidence levels. It is logic to think that the user would not register a birthday in the contact information because the selected contact already had a birthday date set in the address book. The second contact selected by the user was also from the "Friends" group and its city was "New York", but the birthday was not set yet in this case. So, we can see that with the mere selection of the contact, "ContactBirthday" intention could be predicted.

A remarkable point in the experiment performed is that the curves corresponding to the possible intentions of the user are closer when we do not consider the user's preferences than when we do. This fact indicates that although one intention can be more probable than others, the confidence level of the other intentions have a similar (lower) value. The incorporation of the user's preferences allows a better distinction of the actual user intention.

Another interesting fact that can be appreciated in Fig.3 is that the confidence of finished intentions gradually decrements to its original value, as happens with "InviteToMeeting" intention. This is due to the fading function used by the intention graph that gradually decrements by a fixed constant to the strength of the evidence on the performed tasks.

5 CONCLUSIONS

We presented a probabilistic approach that can be used to enhance an interface agent with the capability of detecting a user’s intentions. This approach allows a probabilistic ranking of the possible intentions that the user can be pursuing based on the evidence of the tasks performed by the user on a software application.

Furthermore, our approach combines information about general intentions a user can pursue in
the application with specific information of a particular user regarding those intentions. This is a necessary issue to consider when designing a plan recognizer that aims at being used by interface agents and that is often forsaken by general approaches to the problem.

We performed a set of experiments to test the ability of our Intention Graph for ranking the user intentions based on the evidence about the performed task. With these experiments we conclude that the user's preferences have a fundamental role in the plan recognition process because they allow a better distinction of the actual intention of the user. On the other hand, the fading function in the Intention Graph allows a gradual "forgetfulness" of the evidence that enables our approach to adapt to changes in the intentions followed by the user.

REFERENCES

Fig. 1. Example of an Intention Graph in the calendar management domain

Fig. 2. Example of a Bayesian network built from a set of data examples
Fig. 3. Evolution of the confidence levels of each possible intention a) in the original Intention Graph and b) the Intention Graph with learnt context nodes merged into.
Personalized Detection of User Intentions

Abstract—Interface agents are strategic software components for improving the quality of services to users. To be accepted by users, interface agents need to make useful suggestions always in the context of the user’s intention. The user’s intention should be detected as soon as possible so that the agent can define a way to collaborate with the user. Plan recognition can be applied to identify the user’s goal based on his or her actions in the environment. However, classical approaches to plan recognition fail in two main aspects that make them unsuitable for being used by interface agents: the lack of personalization and the lack of consideration of the transition between different goals pursued by the user. We propose an approach to capture intentions taking into account the variables involved in the application domain that represent the user preferences. Experimental evaluations show us that we have found a way for early detection of intentions.

Keywords—Plan Recognition, Interface Agents, User/Machine Systems, Knowledge personalization and customization.

1 INTRODUCTION

Interface agents [1] are computer programs designed to assist human users in their computer-based tasks in a personalized manner. This kind of agents is able to learn the user’s interests, preferences, priorities, goals and needs aiming at providing him or her with proactive and reactive assistance in order to increase the user's productivity regarding to the use of the application at issue. With the aim of assisting a user of a software application, interface agents not only have to learn the user's preferences and habits regarding the use of the application itself, but should also consider the user’s intention before initiating an interaction with the user.

Considering the status of the user's attention (his or her intention or the goal that he or she is trying to achieve) and the uncertainty about the user's intentions are critical factors for the effective integration of automated services with direct manipulation interfaces [2]. As a result, we must build agents capable of detecting the user’s intention so that they can predict opportune moments for gaining the user's attention.

Most of us have experience in some moment the situation in which a software application foretold what we were intending to do. A typical example of such application is the Office ‘97’s
assistant, Clippy, who interrupted users with phrases such as “It looks like you are writing a letter…” However, Clippy says the same phrase whenever the user starts writing a sentence beginning with the word “Dear ….” Consequently, in these kinds of interventions, Clippy is not capturing correctly the user’s intention.

A correct detection of the user’s intention will be useful for the agent to avoid interrupting the user in an improper moment. Users generally don't want to be interrupted while working on a specific task, unless this interruption is strongly related to the task they are performing. For this reason, most users found Clippy annoying and stopped using it [3]. By considering the user's intention the agent will be able to answer to the user’s requirements always in the context of his or her current goal. For example, if the agent observes that the user is scheduling a work meeting for the following day, the agent can offer to automatically complete the information required and to send an email to each participant of the meeting, providing that it knows the user's preferences about the kind of meeting he or she is scheduling.

With this purpose, plan recognition aims at identifying the goal of a subject based on his or her actions in the environment. Most of previous approaches to the problem of plan recognition fail in two main aspects for being used by interface agents. First, one of the most important problems that an interface agent has to deal with when inferring the user's intention is the uncertainty related to the moment in which the user starts a new plan to achieve a new goal, that is how does the agent become aware that the user has already achieved one goal and started pursuing a new one? This issue is not usually addressed by many approaches to the problem of plan recognition, which consider only one "session" starting with the first observed action and ending when the algorithm recognizes the user's intention. In an interface agent environment, the user will repeatedly start pursuing new goals in the application, with no preplanned behavior. Moreover, the user can even change his or her intention without completing his or her previous goal. This
problem is usually solved restricting the memory of the plan recognizer so that it only considers the most recent tasks performed by user, or it considers each task for only a fixed interval of time and then they are completely disregarded. We present a novel alternative approach in which the strength of the evidence about the user’s actions is lessen over time.

Second, most of previous approaches do not consider adaptation to a particular user. Plan libraries are only domain dependant but user independent, because they are constructed beforehand according to dependencies in the tasks of the domain. Hand-coded plan libraries do not consider that the behavior may vary from one user to another according to the values taken by some variables of the domain. There are variables that define different contexts in which the user may have a different behavior. For example, receiving an invitation for a meeting sets a context for the user’s subsequent behavior, but if that invitation comes from his or her boss, the user may behave completely different.

To face these problems, we propose an approach that will enable an interface agent to perform personalized plan recognition over the tasks that the user executes in a software application. We will not focus on how the agent makes use of the information about the user’s intentions to provide assistance but on providing the agent with a probabilistically ranked list of the most probable user’s intentions in every moment. The main contributions of our approach are the personalization of the plan recognition process and the use of a fading function for evidence to face the problem of transitions between goals pursued by the user. This paper is structured as follows. In Section 2 we present the problem of plan recognition. In Section 4 and 5 we present and evaluate our approach. Finally in Section 7 we present our conclusions.

2 WHAT IS PLAN RECOGNITION?

Consider the following situation. Mary, a secretary, observes her boss, John, opening a
personal information management software application that is well known by both of them. This software application has many tools including managing events, contacts, reminders, home budget, personal diary, lists of favorite web pages, passwords, etc. Next, she observes her boss clicking on the "Add event" button. Now, she is sure he wants to add a new event to the calendar. She knows the steps required to complete this task, and she is able to help him in any moment. John enters the subject of the event, selects the day and starting time and then he clicks the "Add participants" button. After scrolling the list of contacts, he clicks on the "Cancel" button and saves the event. Mary cannot imagine what her boss is intending to do. Perhaps he did not find the contact he was looking for, or maybe he has a meeting and cannot spend more time scheduling the event. But next Mary observes that John clicks on the "Add contact" button, he completes the required information for adding a new contact and saves the changes. With the information the secretary has until this moment, she can think that the participant that the boss was intending to add to the event was not in his contact list so he had to create a new contact and next he will edit the event he had saved before to add the recently added contact as a participant of the event. Moreover, if the secretary has enough knowledge about her boss habits and the events he usually adds to the application, she could be able to predict, for example, that the event will take place in a specific location, at a specific time just by observing the subject, the host and other information he has entered. It may also be possible for her to predict that some member of the office will take part of the meeting based on the type of meeting the boss is scheduling.

The term that has been introduced to describe the process of inferring the intentions of a subject based on the actions he or she performs in an environment is plan recognition. What the secretary in the previous example has done by trying to deduce her boss intention can be considered a plan recognition process. Inputs to a plan recognizer generally are (1) a set of goals the agent expects the user to carry out in the domain, (2) a set of plans describing the way in
which the user can reach each goal, and (3) an action observed by the agent. The plan recognition process itself consists in foretelling the user's goal and determining how the observed action contributes to reach this goal. The set of possible plans the user can follow to achieve a goal constitutes what is called the \textit{plan library}.

There are currently two main approaches to the problem of plan recognition: consistency and probabilistic. Consistency approaches \cite{4} \cite{5} \cite{6} \cite{7} aim at narrowing the set of candidate intentions by eliminating those plans that cannot be explained by the actions that the user performs. Probabilistic approaches \cite{2}\cite{8}\cite{9}\cite{10}\cite{11}\cite{12}, on the other hand, explicitly represent the uncertainty associated to the user's plans and allow to probabilistically rank the most probable user's intentions.

Both kinds of approaches can lead to accurate predictions providing that the plan library is complete in consistency approaches, and that the probabilities are correct in probabilistic approaches. However, probabilistic approaches can find the most probable intention if the observations up to a given moment enable more than one possible intention, while consistency approaches cannot select between them and have to wait for a single consistent explanation to make a prediction. See \cite{13} for a complete review of plan recognition for interface agents.

There are two main aspects that make classical approaches to plan recognition unsuitable for being used by interface agents. First, the agent should deal with transitions and changes in the user intentions. The agent usually will not be certain about the moment in which the user starts a new plan to achieve a new goal. Classical approaches do not consider this problem and they restrict the plan recognition process to only one "session" that starts with the first observed action and ends when the algorithm recognizes the user's intention. In an interface agent environment the user not only will repeatedly start pursuing new goals in the application, with no preplanned behavior, but also he or she may change his or her intention without completing the
previous one. Those approaches which do consider this problem restrict the memory of the plan recognizer so that it only considers the most recently performed tasks, or they consider each task only for a fixed interval of time, completely disregarding them after that “lifetime”.

Second, user preferences have a fundamental role for an interface agent and should be considered in the plan recognition process. Considering the user’s preferences in the plan recognition process is an important factor that an interface agent should regard because the behavior of the user in a specific situation is usually determined by his or her preferences related to that situation. In this context, a situation is described by the information the user is handling to achieve his or her intention. In the calendar scheduling domain, for example, the situation related to a new event addition may be described by the event’s date, time, participants, place, subject, etc. These variables of the domain strongly influence the user behavior. For instance, if the user adds a new event which time is within his or her working hours, there is a high probability of the event taking place in the office.

The process of adapting a computer application to the needs of a specific user is called personalization and takes advantage of knowledge acquired from the analysis of the user's behavior and data. In contrast to customization which is a user-initiated and user-driven process, personalization is system-initiated and system-driven and thus requires the system to monitor the user's behavior in order to adapt automatically. The fundamental purpose of personalization is the user's satisfaction, and it is motivated by the acknowledgment that a user has needs and preferences, and meeting them successfully is likely to lead to a satisfying relationship with him or her.

The importance of personalizing the plan recognition process by considering user preferences has been identified previously in [16] where decision trees were used to learn classes of situations based on the user’s actions. This approach runs the plan recognizer on a set of input
examples from a typical user, and then gathers statistical data based on the results of running the plan recognizer on the entire observable behavior in each episode. The main problem of using decision trees to "classify" the intention of the user according to some attribute of his or her actions is that they can only assign one possible intention given those attributes, completely discarding other options.

In the next section we describe some related work in the area of plan recognition. Then, in Section 4, we present our approach to provide an interface agent with a probabilistic ranking of the most probable user’s intentions which takes into account the user’s preferences.

3 Related Work

The prediction of the user’s goal is an inherently uncertain task. Therefore, it is desirable to have a knowledge representation capable of dynamically capturing and modeling the uncertainty of human-computer interactions. Non-probabilistic approaches [4][5][17] are weakened by the fact that they cannot decide to what degree the evidence supports any particular hypothesis about the user’s goal. This is an important issue to consider for the agent to be able to rank different possible explanations supported by the set of performed actions. Markov models and Bayesian networks are two of the most important representations for dealing with this kind of information.

Mott et al. [18] performed a set of empirical studies on two families of probabilistic goal recognizers, n-gram models and Bayesian networks and conclude that probabilistic approaches can perform goal recognition that is accurate and incrementally converging.

The main advantage of n-gram models (Markov chains) is that they can be derived from data without any previously known structure. However, the Markov assumption is unrealistic in our domain. A major disadvantage of higher order Markov chains (n-grams, for n>1) is the exponential growth in the state space when we increase the order of the model to capture more
than very short term memory dependencies in the sequences. Blaylock and Allen [19] explored the use of n-gram models to perform statistical goal recognition. Specifically, they built two experiments in the Linux domain using unigram and bigram models, respectively. Then they defined the problem of goal recognition as a classification task where given an observed sequence of $k$ instantiated actions observed until a given moment, the algorithm searches for the most likely instantiated goal.

Bayesian networks have also been successfully used in previous approaches to plan recognition. Charniak and Goldman [10] built the first probabilistic model of plan recognition. They use a quantifier-free first order language as representation and belief nets for plan inference. The random variables in the belief net are propositions, whereas the root nodes are hypotheses about agent's plan. Charniak and Goldman apply their plan recognition system in the area of story understanding. They apply a set of rules to a sequence of observations using plan knowledge, to automatically generate a Bayesian Network. Then, after incorporating prior probabilities, this network supports the selection of possible explanations of observed actions using the posterior probability of each hypothesis, computed by propagating the values from the evidence in the net. This approach assumes that the plan library is represented in a quantifier-free first order language. The network is built from rules over that representation, but it is not adapted to a particular user.

Brown [8] also used Bayesian networks to build a user profile that enables the detection of a user's intentions considering the utility of offering assistance. This approach is based on the belief that what the user is intending to do in an environment is the result of the stimulus (i.e. the events) that occurs in the environment, and of the goals he is trying to reach as a reaction to those stimuli. Goals are made of a set of actions, with pre and post conditions. Preconditions include both events that are directly observable in the environment (such as the user clicking in a button
labeled “Add event”) and events that are indirectly observable (such as increase cognitive load of the user). These preconditions cause the user to intend to achieve a goal. Goal abstractions facilitate the design and detection of higher level goals in the search of lower level goals. Evidence can be added and removed easily and intuitively (in the form of pre and post conditions) as the user interacts with the system. Pre and post conditions for goals and actions can be explicitly indicated, and keyhole plan recognition is made easier by enumerating explicitly the actions that make up goals. Brown assumes that the user is never observed to perform a goal explicitly. Although this assumption is correct, the user's feedback can be considered by the agent to know in a given moment the user intention.

More recently, Goldman, Geib and Miller [11][12] presented a probabilistic theory of plan recognition based on plan execution. They use a simple hierarchical (task decomposition) representation of plans that provides recipes for achieving goals. In their model of plan execution, initially the user has a set of goals and chooses a set of plans to execute to achieve those goals. The set of plans chosen determines the set of pending primitive actions. Then the user will repeatedly execute one of the pending actions and generate a new set of pending actions from which further action will be chosen. The new pending set is generated from the previous set by removing the last executed action and adding the actions that become enabled by it execution. Actions become enabled when their required predecessors are completed. The algorithm first computes the complete and covering set of possible explanations; second, it computes the probability of each of the explanations; finally, it computes the conditional probability of the given goal on the basis of the probability of the explanations. A disadvantage of this approach is that the number of explanations can grow exponentially in the number of root goals that share a common unordered prefix of actions.

When a plan recognizer is to be adapted to a particular user of a software application, a proper
model of his preferences is crucial because preferences are the reasons why a user acts in a particular way when facing a certain situation [16]. For example, a user of a scheduling application can prefer organizing events in his house when the participants are his or her friends or when it takes place in the afternoon, but might prefer the club for events which take place in the morning. Ignoring the user's preferences hinders the plan recognition process, especially if the behavior of the user deviates from what is typically to be expected in a domain.

Machine learning techniques are the most widely used method to build user models implicitly. The main reason of the popularity of these techniques is that user models can be obtained incrementally, usually without intervention of the user, by observing the user while he performs his tasks.

Besides Bauer’s approach described in Section 2, other approaches have been proposed for acquiring domain knowledge by probabilistic reasoning. Chen et al [20], for example, presented a method to model and inferring user's intention using data mining. They make the distinction between action intention (low level, such as mouse click, keyboard typing and other basic actions performed on a computer) and semantic intention (what the user wants to achieve at high level, which may involve several basic actions on a computer to accomplish it). They work in the domain of a web browser, and mainly focus on predicting action intention based on the features they extracted from the user interaction such as user's typed sentences and viewed content. A modified Naïve Bayes classifier is used to model the user's action intention on a computer. The model can be trained incrementally and used to predict the user's next action. However, they only focus on the prediction of action intentions, i.e. the prediction of the next action of the user based on the document he or she is browsing or what he had typed in a web browser. Plan structure is not considered in this approach.

Maragoudakis et al. work [20] is based on inferring intentions for actions from questions in an
educational dialog engine. They use automatic encoding of the semantic model of the application, based on learning Bayesian networks from past user questions. Both the structure of the networks and the conditional probability distributions are computed automatically from dialog corpora. This framework is Bayesian in that they start from a causal theory of how the agent's mental state causes its plan, executing its plan causes activity, and they reason from observed effects to underlying causes.

In the next Section we present our approach for providing an interface agent with a probabilistic ranking of the most probable user’s intention.

4 Probabilistic Ranking of User Intentions

In general, a user performs a set of tasks in order to achieve a higher level goal. For example, to arrange a dinner with his or her friends, the user has to look at the calendar for a convenient date, he or she has to create a new event in his or her agenda and has to write an invitation mail to each participant of the meeting. Therefore, when the user performs a task, the agent should consider the set of possible intentions associated to it to find a way to collaborate with the user. The prediction of the user's intention is an inherently uncertain task. Therefore, we look for a knowledge representation capable of capturing and modeling dynamically the uncertainty of human-computer interactions.

In our approach, we represent the set of intentions the user can have in the application domain with an Intention Graph. We define an Intention Graph to be a representation of the context of execution of tasks and it is materialized by a Bayesian network. The context is viewed as the set of tasks that the user has performed recently, and will influence the confidence that the agent has in any given intention that the user may have.

Bayesian Networks are a probabilistic knowledge representation used to represent uncertain
information. Bayesian networks are directed acyclic graphs representing both the conditional dependencies and independencies between elements of the domain. Knowledge is represented by nodes called random variables and arcs representing causal relationships between variables. Each variable has a finite set of mutually exclusive states. Nodes without a parent node have an associated prior probability table. On the other hand, the strengths of the relationships are described using parameters encoded in conditional probability tables (CPTs).

Bayesian networks are used for calculating new probabilities when some particular information becomes available. The information is called evidence and will be in the form "A=a", where A is a variable and a is a state of A. In our case, we will have new evidence every time the user performs a task in the software application. Therefore, evidence will be in the form “AddContact=true” meaning that the user performed “AddContact” task.

The general setting for probabilistic inference is that we have a set V=V₁, V₂, ..., Vₖ of propositional variables and we are given as evidence that the variables in a subset E ⊆ V have certain definite values, E=e. In the probabilistic context, we want to compute the conditional probability distribution p(Vᵢ|E), for each variable Vᵢ given the evidence. This process is called probabilistic inference.

By making use of probabilistic inference and considering the set of tasks performed by the user as evidence, we will be able to compute the probability that the user is pursuing any given intention modeled in the Intention Graph. Moreover, if the user explicitly declares his or her intentions, we will be able to probabilistically assess the tasks that he or she has to perform to achieve his or her goal.

4.1 Building an Intention Graph

In our Intention Graph variables correspond to goals that the user can pursue in the application
domain and to tasks that the user can perform in the application to achieve those goals. The two possible states of these variables are true and false. A true value in a variable representing a goal indicates that the user is pursuing that goal. On the other hand, a true value in a variable representing a task indicates that the user has performed that task in the application. We call confidence level to the probability of a variable being in state true. Notice that we will not have direct evidence about the goal the user is pursuing unless the user makes an explicit declaration of his or her intention. Evidence on a task node will be set when the user interacts with a widget in the application GUI that is associated to the execution of the corresponding task. Finally, our Intention Graph includes a third kind of variable: context variables. This kind of variables will be used to personalize the intention detection process and will be explained later in Section 4.4.2.

In order to construct the graph, an expert in the application domain should first add a node for each task and intention in the domain and then draw arcs according to existing relationships among tasks and intentions. Missing arcs encode probability conditional independence assertions, as no dependency exists among unrelated nodes. Consider that the tasks that the user performs are caused by what he intends to do in the application, the direction of arcs in our Intention Graph is always from goals to tasks. Relations among tasks are also allowed and indicate that the execution of one task influence the execution of other task.

Once the structure of the Intention Graph is built, the domain expert should assess the strength of the relations between variables my means of conditional probabilities tables in each node with parents in the graph and a priori probabilities for nodes without parents. Although the probabilities selected will have a strong influence in the quality of the Intention Graph, they can be automatically updated later by using the user feedback about his or her intentions, as we will detail in Section 4.4.
4.2 Example

To illustrate our proposed approach we will use an application called Agenda 4.0 that provides various functions for managing personal information such as contacts, tasks, events, and email.

The calendar domain is a well known domain in which interface agents can be applied. An interface agent in this domain can assist users with tasks such as suggesting meeting places, dates and times for a new event the user is scheduling, suggesting the participants for a meeting, notifying the user when an event is about to happen and warning about overlapping events and about an unusual event scheduled by the user. As stated before, to provide this functionality, the agent should be aware of the set of tasks that the user is performing because it gives the context in which the user is putting his or her attention. This context will enable the agent to infer the user's intention and it will be able to provide assistance to the user, or it can collaborate with the user to fulfill the intention he or she has.

The Intention Graph we designed for this application is shown in Fig.1. Nodes labeled with black background and white text correspond to intention nodes and nodes labeled with white background and black text correspond to task nodes.

As we can see, we identified six intention nodes: "InviteToMeeting", "SendMailToContact", "RegisterBirthday", "ContactBirthday", "EventBirthday" and "AddMissingContactGroup". "InviteToMeeting" reflects the intention of the user of arranging a meeting with a selected contact. On the other hand, "SendMailToContact" reflects the intention of the user of sending an email to the contact he has selected in the application. "RegisterBirthday" is a goal that can be achieve either by registering the birthday of a selected contact in the personal information of that contact ("ContactBirthday" intention) or by creating a new event to remember about that important date ("EventBirthday"). Finally, "AddMissingContactGroup" is used to detect when
the user has to add a new contact group to the agenda to be used afterwards as the group of a
given contact. Although a simple example, in Section 5 we show how beliefs on the intentions
modeled in the Intention Graph are updated as a user interacts with the application.

4.3 Fading Past Observations

Recall the example of the secretary presented in Section 2. What should Mary, the secretary,
think if before completing the intention she believes her boss has, John receives a telephone call
and Mary observes that he makes a new entry in his personal diary in the application? Should she
consider that her belief was wrong? John can yet continue editing the event after saving the new
entry in the diary. Or perhaps John has abandoned his previous goal and started a brand new one.
This example shows that plan recognizers must be careful not to over-commit themselves to one
possible intention. Especially in a domain where users can change they current goal rapidly, it is
often a good idea for the plan recognition system to be a little "forgetful".

As stated in Section 2, most of previous approaches to the problem of plan recognition do not
consider the uncertainty related to the moment in which the user starts a new plan to achieve a
new goal. Those which consider this issue limit the memory of the plan recognizer by making
evidence to be present in a fixed interval of time and then completely disregarding it. We take a
different approach in which evidence is gradually forgotten.

In Bayesian networks terms, evidence is a collection of findings on variables. A finding may
be hard or soft. A hard finding specifies which value has taken a given variable. Findings on the
values taken by the variables introduced before are of this kind. A soft finding, on the other hand,
specifies the probability distribution over a variable’s possible values. Hard evidence is a
collection of hard findings and Soft evidence is a collection of soft findings.

In our approach, we adopt the concept of soft evidence to fade the evidence we entered to the
Bayesian network as the user performs further tasks. To do so, we use a fading function \( F(\cdot) \) to gradually forget the tasks performed by the user, as shown in Eq. 1.

\[
P(t = \text{true}) = P_{\text{prev}}(t = \text{true}) - F(\cdot) \quad \text{Eq. (1)}
\]

That is, the current probability that task \( t \) was performed, \( P(t=\text{true}) \), is computed by subtracting from the previous probability assigned to task \( t \), \( P_{\text{prev}}(t=\text{true}) \), a value assigned by the fading function \( F(\cdot) \). Evidence on every node is faded according to this function until they reach their original value, i.e. until the confidence of a given node is lower than the value that it would have if we wouldn't have observed the execution of the corresponding task in the application.

By gradually forgetting past observations, the agent not only will be able to manage changes in the user’s goal, but also will let it to forget the execution of noisy tasks, that are tasks that do not belong to the main goal the user has in a given moment. In our example, the boss can check the current date for its own sake, and not because it is part of the plan he is following in that moment.

Fading functions can be any function that, given the Intention Graph and the evidence on tasks performed so far, gradually decrements the certainty of the evidence according to some heuristic [15]. For example, we can decrement current evidence by a fixed factor \( 0 \leq \Delta \leq 1 \) every time the user performs a task in the application. This way, for all nodes with evidence \( t_i \) we will update the probability distribution of its evidence according to Eq. (2)

\[
P(t_i = \text{true}) = P_{\text{prev}}(t_i = \text{true}) - \Delta \quad \text{Eq. (2)}
\]

This simple fading function will allow the agent to disregard, after some actions performed by the user, a previously performed noisy action or a previously pursued goal. This function, however, might have the problem of rapidly forgetting evidence in tasks that actually contribute
to the user’s current goal. This problem can be solved using another function that, for example, keeps almost intact the evidence of some number of tasks and then quickly decrement up to the original value of the node (without evidence).

4.4 Personalizing the Detection of the User’s Intentions

When a plan recognizer is to be adapted to a particular user of a software application, a proper model of his or her preferences is crucial because preferences influence the way that a user acts when facing certain situations [16]. In our example, the secretary should learn that her boss prefers organizing events in his house when the participants are his friends or when it takes place in the afternoon but that he prefers the club as the meeting place for meetings taking place in the morning. Ignoring the user's preferences hinders the plan recognition process, especially if the user’s behavior deviates from what is typically expected. So, an agent provided with plan recognition and user modeling capabilities can detect patterns of behavior that are particular to each different user of the application.

In our approach, we personalize the Intention Graph initially constructed by the domain expert in two ways: (1) by adapting the probabilities given new user experiences (Section 4.4.1) and (2) by learning new relations between the attributes of the tasks that influence the user’s intentions (Section 4.4.2).

4.4.1 Adaptation of Probabilities in the Intention Graph

As stated above, by using Bayesian networks inference mechanism the agent will be able to update its belief in any given goal related to the tasks performed by the user. This belief update is made using the probabilities set by the domain expert who designed the Intention Graph. However, when constructing a Bayesian network the domain expert may be uncertain of the correctness of the conditional probabilities chosen. Nevertheless, when the agent is at work it will repeatedly get new experiences of tasks performed by the user when he or she has some
particular goal. It would be desirable that the agent could learn from these new cases to adapt the probabilities set by the domain expert who has designed the Intention Graph. This adaptation process will make it possible to model the way of acting of a particular user.

For this task we take a statistical on-line learning approach. While the user interacts with the application, the agent “remembers” the tasks performed by the user. Then, when the agent infers that the user is pursuing certain goal, it adapts the probabilities of the intention graph to reflect this new experience. The agent will believe that the user has already completed the current goal(s) when a certain threshold is exceeded. This threshold should be empirically determined for each application domain. Explicit feedback can also be considered if the agent believes that it can ask the user whether he is trying to achieve a given goal or not, but this decision is out of the scope of our work.

The statistical method we used to update the CPTs of the variables in the network is called Fractional Updating and was proposed by Jensen [22]. The statistical task is to gradually modify the estimates of the parameters with the new experiences entered. To do so, besides the CPT for each node, we keep an experience table. The experience table of a variable represents the experience counts of the parent configurations. For example, in the Intention Graph in Fig.1, the experience table for the node labeled “Events” will have two entries, one indicating the number of times we observed the execution of “Events” task when the user intended to register an event to remember about a birthday of a specific contact (“Event-Birthday” intention) and the other indicating the number of times we observed the execution of “Events” task when the user was not pursuing “EventBirthday” intention.

Let $n_k = P(N=k)$, the probability of node $N$ being in state $k$, $y_k = P(N=k|e)$ the probability of node $N$ being in state $k$ given the current evidence in the Bayesian network, $conf$ a given configuration of the parent nodes of $N$, $z$ the probability of such configuration given the current evidence and
the previous experiences count for $n_k$. We update the CPT for node $N$ being in its state $k$ for a given parent configuration $conf$, according to Eq. 3.

$$P(n = k \mid conf) = \frac{n_k + z \cdot y_k}{n_{\text{prev}} + z} \text{ Eq. (3)}$$

For further information about Fractional Updating, please refer to [22].

### 4.4.2 Learning New Relations in the Intention Graph

A further adaptation that can be done to the Intention Graph built by the domain expert to learn new relations that may arise between the attributes of the tasks performed by the user and the intention nodes in the Intention Graph.

For example, in *Agenda 4.0*, the user can select a contact from the address book with the objective of sending this contact a mail or with the objective of scheduling a meeting with this contact, as shown in the Intention Graph presented in Fig.1. The Intention Graph constructed manually by a domain expert will enable the agent to rank which of these two goals is more probable, given that the user has selected a contact from his or her address book. However, the information about the selected contact can be relevant in discerning which goal the user actually has.

To consider this information, we introduce to the definition of our Intention Graph, the concept of *traceable nodes*. A traceable node is a node of the Intention Graph in which we want to register the values taken by some attributes of the corresponding task when it is performed by the user with. These attributes will be modeled as new variables that represent the context in which the user performed the task corresponding to a traceable node and will be used to find new relations between these variables and the nodes in the Intention Graph.

In the example above the task corresponding to the selection of a contact in the address book is a traceable node. The designer of the Intention Graph should decide which attributes of this task
are interesting (for example, the city in which the contact lives or the group to which the contact belongs) for which set of intentions (in the example, sending a mail to the selected contact or scheduling a meeting with him or her).

Every time the user performs a task corresponding to a traceable node, the agent will observe the values taken by the attributes of the task (for example, the selected contact is from New York and belongs to his group of friends). Then, the agent will continue observing the user until it can infer which his or her intentions are and will save the experience in an interaction history. Each experience will be in the form

\[
\langle \text{attribute}_1, \text{attribute}_2,...,\text{attribute}_n, \text{intention}_1,...,\text{intention}_k \rangle
\]

where \( \text{attribute}_i \) is the value taken by the attribute \( i \) and \( \text{intention}_j \) is true if the agent infers that the user was pursuing intention \( j \) or false otherwise.

This database of experiences is then used by the agent to run both a batch learning and a parametric learning algorithm to find relations between the attributes themselves and between the attributes and the intentions. In our implementation, we use the PC algorithm [23] for learning the structure of the network from the database of experiences and the EM algorithm [24] for learning the probabilities in the CPTs. Fig.2 shows an example of a network learnt from experiences in the scheduling application.

In this machine-learned Bayesian network, variables representing attributes are incorporated to the network in the form of a new kind of variables, context nodes (gray nodes correspond to this kind of nodes nodes), and variables representing intentions as intention nodes. The learnt network is then merged with the Intention Graph to incorporate this knowledge to the detection of the user’s intention.
5 Experimental Results

To perform our experiments, we selected the calendar management domain for two reasons. First, because it is a well known and easy to communicate domain in which interface agents can be applied. Second, because of the richness in the number of variables involved which make it a non-trivial domain. An interface agent in this domain can assist users with tasks such as suggesting meeting places, dates and times for a new event the user is scheduling, suggesting the participants of a meeting, notifying the user when an event is about to happen and warning about overlapping events and about an unusual event scheduled by the user. A complete development of such an agent is out of the scope of this work. We are concerned only in providing an ordered ranking of the possible goals the user can pursue.

In this section, we present two scenarios. The first scenario aims at evaluating the influence that contextual information representing the user’s preferences has in the prediction of the user’s intentions. The second scenario shows how the Intention Graph, due to the fading function, reacts when the user performs noisy tasks.

5.1 Scenario 1

With the objective of showing the influence of the user's preferences in the detection of the user's intention, we select a scenario observed from the interaction of a regular user of the scheduling application in which he utilized the scheduling application to organize a meeting with some contact in his address book and then selected another contact to register his or her birthday. To achieve these goals, the user performed the following sequence of tasks: <"SelectContact", "AddContactToMeeting", "SelectContact", "EditContact", "PersonalInformation", "EnterBirthday">.

We recorded the confidence levels for all intentions both in the Intention Graph containing the
user's preferences information and in the same Intention Graph without context nodes (the one owned originally by the agent). Figure 3.a shows the evolution of the confidence levels of each possible intention in the original Intention Graph without context nodes. In the first time slice, we show the a priori probabilities of each intention when the user did not perform any tasks in the application. "SendMailToContact" is the more probable intention, while "ContactBirthday" is the least probable one. When the user performed the first task, "SelectContact", the ranking remained unchanged, although there was a little increment in those intentions that contained this task. Then the user performed "AddContactToMeeting" and "InviteContactToMeeting" became the most probable intention. With the following set of tasks performed by the user, "ContactBirthday", "RegisterBirthday" and "EventBirthday" started gaining confidence while the other intentions decreased their confidence level. "AddMissingContactGroup" confidence level remained unchanged along this session because it is "disconnected" from the tasks performed by the user. The agent in this case considered a threshold level with a value of 0.7 to believe in the intention pursued by the user; it predicted the first intention after observing the second performed task and the second intention after observing the fifth task. If we consider the number of tasks in each intention, it needed both tasks to be performed to detect "InviteToMeeting" intention, and three tasks out of four to detect "ContactBirthday" intention.

Fig.3.b shows the same scenario but performed using the Intention Graph with context nodes merged. The first "SelectContactTask" was performed when the user selected a contact from the "Friends" group, living in "New York", and that already has the birthday registered. We can see that the confidence level for "InviteToMeeting" is higher only with the first task performed. We can also see that the confidence levels of the remaining intentions were dramatically lowered. It is logic to think that the user would not register a birthday in the contact information because the selected contact already had a birthday date set in the address book. The second contact selected
by the user was also from the "Friends" group and its city was "New York", but the birthday was not set yet in this case. So, we can see that with the mere selection of the contact, "ContactBirthday" intention could be predicted.

A remarkable point in the experiment performed is that the curves corresponding to the possible intentions of the user are closer when we do not consider the user's preferences than when we consider them. This fact indicates that although one intention can be more probable than others, the confidence levels of the other intentions have a similar (low) value. Therefore, the incorporation of the user's preferences enables a better distinction of the actual user intention.

Another interesting fact that can be appreciated in Fig.3 is that the confidence of finished intentions gradually decrements to its original value, as happens with "InviteToMeeting" intention. This is due to the fading function used by the intention graph that gradually decrements by a fixed constant the strength of the evidence on the performed tasks.

5.2 Scenario 2

In this second scenario we test how the Intention Graph reacts when the user performs a “noisy” task and follows two intentions at the same time. In this scenario, the user forgot greeting one of his friends living in Buenos Aires about his birthday, so he intends to create a remainder in the application for the next year, register the birthday date in the personal information of his friend, and write an email to congratulate him and to apology for his absentmindedness. The ordered set of tasks performed in this scenario is <“Events”, “AddContactGroup”, “New Event”, “SelectContact”, “EditContact”, “PersonalInformation”, “EnterBirthday”, “SendMail”>. The intentions followed by the user that lead him to perform those tasks are “EventBirthday” which is interrupted by the execution of “AddContactGroup”, “ContactBirthday” and “SendMailToContact”.
Fig. 4 shows the evolution of the confidence level for the intentions in the Intention Graph as the user performs the mentioned tasks. When the user changes to the events tab in the application, the confidence for registering a birthday by creating an event increases near to 0.78. Then the user performs a noisy task and the confidence level of this intention decreases, but when the user resumes the previous intention by performing “NewEvent” it is increased again. However, we should notice that the increment in the confidence after resuming the intention that was interrupted by the noisy task is not very significant. This is due to the fact that the noisy task caused a decrement in the strength of the evidence we had on the “Events” task.

The selected contact in the fourth task was part of the “Friends” group, lived in “Buenos Aires”, was a “System Engineer” and did not have a birthday registered in the application. When the user selected this contact for edition, “ContactBirthday” becomes the most probable intention. However, due to the characteristics of the selected contact, “SendMailToContact” also increases its confidence level. As the user continues performing tasks in the application to achieve his intention of registering the birthday of the selected contact, the confidence level on “SendMailToContact” is gradually decreased. However, when the user finally performs “SendMail” task, it gets a value above 0.9.

6 COMPARISON WITH GOLDMAN, GEIB AND MILLER'S APPROACH

Our model is situated within probabilistic approaches to the problem of detecting the user's intentions and combines both keyhole and intended plan recognition. Keyhole plan recognition is accomplished by observing the actions performed by the user in a non-intrusive manner. Intended plan recognition is incorporated to the model by allowing the user to provide feedback about his intentions. This explicit feedback will allow the agent to adapt the intention model to a particular user of the application.
From the plan recognition approaches presented in Section 3, the one that is most similar in spirit to our model is the model for plan recognition proposed by Goldman, Geib and Miller [11][12]. This approach handles partially-ordered plans, interleaved plans, considers the effect of context in plan choice and is able to manage the execution of actions for its own sake. It also supports missing observations, and noisy tasks that do not contribute to the main goal of the user.

Their approach is both consistency and probabilistic and is based on plan execution. Plans are represented as a hierarchy (task decomposition) providing recipes for achieving goals. Fig. 5 shows an example plan library for a hypothetical space station example, as presented in [11]. Tied arcs from a parent node to a child node represent “and” nodes, while untied child represent “or” nodes, or alternative plans to achieve the (sub) goal represented by a node. For example, it is possible to increase the power (“increase-power”) either by generating power (“gen-power”) or by using less power (“lower-power-use”); to generate power, it is need to open panel 1 (“open-p1”) and to start generator B (“start-gen-B”). Horizontal arrows represent temporal restrictions. For example, before starting generator B, it is needed to open panel 1. Dashed lines represent the influence of context on the possible goals. For example, the existence of an oxygen drop (“o2-drop”) influences the fact that the user will intend to raise the level of oxygen (“raise-o2-level”).

In Goldman’s et al. model of plan execution, the user initially has a set of goals and chooses a set of plans to execute to achieve those goals. The set of plans chosen determines the set of pending primitive actions. Then the user will repeatedly execute one of the pending actions and generate a new set of pending actions from which further action will be chosen. The new pending set is generated from the previous set by removing the last executed action and adding the actions that become enabled by it execution. Actions become enabled when their required predecessors are completed.
They define an explanation of a set of observations as a minimal forest of instances of plan trees with expansions chosen for “or” nodes sufficient to allow an assignment for each observation to a specific basic action in the plan. The algorithm first computes the complete and covering set of possible explanations; second, it computes the probability of each explanation; finally, it computes the conditional probability of the given goal on the basis of the probability of the explanations.

In order to make Goldman’s et al. model probabilistic, it is necessary to specify probabilities for a restricted set of hypothesis. First, we have to specify the probabilities for the top-level goals which may be a priori, or conditioned on the context. In the example in Fig. 5, we have to specify probabilities for “increase-power”, “raise-o2-level” and “raise-temp”. Then we have to set the probabilities that the user will chose a particular method when attempting to achieve a goal. In the example, the probability that the user will choose to perform “gen-power” or “lower-power-use” when he aims at increasing the power, and “lower-o2-use” or “gen-o2” when he intends to achieve “raise-o2-level”. Finally, we have to detail the probabilities that the user will chose a particular primitive action from a set of pending primitive actions.

Based on these probabilities, the probability of a given explanation is computed by multiplying together the priors for each goal, the probability of the expansion choices at each “or node”, and the probability of the observed actions being chosen from the pending sets:

$$p(\text{explanation} | \text{observations}) = \prod_{i=0}^{l} p(\text{goal}_i) \prod_{j=0}^{J} \frac{1}{|\text{Choice}_j|} \prod_{k=0}^{K} \frac{1}{|\text{PS}_k|}$$

where the first term is the probability of the user's goal, the second term the probability that the user chooses the particular plans to achieve those goals, and the final term captures the probability that the given observations occurred in the specified order. In the formula presented
above, it is assumed that each method is equally probable given its parent goal and that actions are equally likely to be chosen from the pending set.

Having the probability of each explanation given the observations, the conditional probability of a specific goal \( g \) is computed by summing the probability mass associated with explanations that have the goal present and dividing it by the total probability mass of all the explanations:

\[
p(g|observations) = \frac{\sum_{e}^{Exp} p(e, observations)}{\sum_{e}^{Exp} p(e, observations)}
\]

where the denominator sums the probability of all explanations for the observations, and the numerator sums the probability of the explanations in which goal \( g \) occurs.

### 6.1 Experiment configuration

In order to compare the behavior of our model with Goldman’s et al. model we will use the sample plan library for the hypothetical space station example in Fig. 5. We will not consider the “context” nodes (“EVA-prep” and “o2-drop”) for simplicity. The Intention Graph we build for this plan library is shown in Fig. 6.

For both models we performed the following sequence of tasks: <“open-p1”, “check-temp”, “start-gen-B”, “raise-temp-set”, “open-p1”, “open-p2”, “shutoff-x2”, “start-gen-B”, “start-o2-gen”, “check-temp”, “shutoff-x1”>. This complex task sequence contains multiple interleaved goals and noisy tasks. With the first four tasks, the user’s goals are both “increase-power” and “raise-temp” goals. Then the user intends to achieve both “increase-power” and “raise-o2-level” and he or she performs a noisy “check-temp” in the way of achieving those goals. We analyze the behavior of the probabilities for the top level goals. Fig. 7 show the results obtained for Goldman, Geib and Miller model and Fig. 8 shows the results obtained with our approach.

We are interested in comparing the behavior of the curves obtained for each intention, rather
than the probability values obtained for each intention at each time step. The first difference to notice is that in Goldman’s et al. approach, the probability values for all intentions before any task is performed is 0 while our model is able to predict each goal with an a priori value. This reflects the fact that when no evidence is observed the user is more likely willing to archive some goals rather than other goals. When the user performs “open-p1”, probabilities for both “raise-o2-level” and “increase-power” grow in both approaches. However, in Goldman’s et al. approach, this probabilities remains constant until the user performs “raise-temp-set”. This is due to the fact that, with the evidence available until this moment, the model cannot decide which of the top level intentions the user has since both “open-p1” and “start-gen-B” are part of both candidate intentions. In our model, on the other hand, “increase-power” is always more probable due to the a priori probabilities encoded in the model. The variations in the values of these intentions are due to the fading function applied in our model. When the user performs “raise-temp-set”, in Goldman’s et al. approach the system starts to believe that “increase-power” is more probable than “raise-o2-level” because it fails to observe “start-gen-B”. On the other hand, in our model the probabilities for both intentions decrease because of the fading function.

When the user performs “open-p1” again, both intentions (“raise-o2-level” and “increase-power”) gain confidence in both models. In our model, this reflects the fact that evidence on “open-p1” is reinforced. On the other hand, in Goldman’s et al. approach the increment in the confidence in both intentions occurs because they create a new set of explanations containing new instances of plans containing this task.

When “open-p2” is performed both models considerably increase their belief in “increase-power”, since “open-p2” is a task that only contributes to “increase-power”. However, in this situation there is an important difference between both approaches. Goldman’s et al. approach will maintain almost constant its belief in “increase-power” until the user performs the last task
(“shutoff-X1”). Instead our model gradually decrements its belief in “increase-power” since the user continues performing tasks that contribute to other intentions and will increment its belief in “increase-power” again when the user finally performs “shutoff-X1”.

In the example, “check-temp” was performed as a noisy task. With the observation of this task, Goldman’s et al. model will believe the user is pursuing “raise-temp” intention and will remain expecting the user to perform “raise-temp-set”, a fact that will not occur. Our approach does not have this problem since evidence in “check-temp” will be gradually forgotten by the model due to the fading function.

6.2 Discussion

Both approaches are able to handle partially-ordered plans, interleaved plans and consider the effect of context in plan choice. One advantage of Goldman’s et al. model with respect to our model is that it supports “negative evidence”: the confirming or disconfirming effect of failing to see some action. In the scenario presented above, when the user performs “raise-temp-set”, it fails to observe the execution of “start-o2-gen” that would contribute to “raise-o2-level” goal, so it decrements its belief in this intention and increments it belief in “increase-power” which contains “open-p1” and “start-gen-B” but not “start-o2-gen”.

Although not shown in the example presented above, both approaches allow to model the influence of context in the intention the user is pursuing. However, in Goldman’s et al. approach contextual relations are fixed while in our approach they are learned from previous experiences. In the same way, our model enables adaptation of probabilities given the user's feedback.

An important disadvantage of Goldman’s et al. approach is that the number of explanations can grow exponentially in the number of root goals that share a common unordered prefix of actions. These “shared leaders” appear in domains where many different plans for different goals
have a common prefix of actions, and have a significant impact on the algorithm's run-time. Plan libraries without ordering constraints represent the worst case for the algorithm performance. In our proposal, there is also an increment in the run-time when we add context nodes representing the user’s preferences. However, this increment in run-time is compensated by the fact that the user’s intentions are detected earlier using this contextual information.

7 CONCLUSIONS

We presented a probabilistic approach that can be used to enhance an interface agent with the capability of detecting the user’s intentions. This approach enables a probabilistic ranking of the possible intentions that the user may have based on the evidence of the tasks performed by the user.

Furthermore, our approach combines information about general intentions that any user can achieve in the application with specific information of each particular user regarding those intentions. This is a necessary issue to consider when designing a plan recognizer that aims at being used by an interface agents and that is often not addressed by general approaches to the problem of plan recognition.

We performed a set of experiments to test the ability of our Intention Graph to rank the user’s intentions based on the evidence about the performed tasks. With these experiments we conclude that the user's preferences have a fundamental role in the plan recognition process because they enable a better distinction of the user’s actual intention. On the other hand, the fading function in the Intention Graph enables a gradual "oblivion" of the evidence that enables our approach to adapt to changes in the user’s intentions.

We compared the behavior of our framework to that proposed by Goldman’s et al. This model has a different behavior compared to our model when facing a set of tasks that can be performed
to achieve more than two goals. For example, suppose that goal G1 is achieved by performing tasks T1 and T2 and goal G2 is achieved by performing tasks T1, T2 and T3 and the system observes tasks T1 and T2. By using the model proposed by Goldman’s et al. we are not able to predict which goal the user is pursuing because we have two explanations for the performed tasks so each one will have the same probability of 0.5. Our model, on the other hand will rank goals G1 and G2 according to their prior probabilities and the propagation of the evidence introduced in the Bayesian network. If task T4 is observed next, we can remark another difference between both approaches. Goldman’s et al. model will increase the probability of goal G1 as it failed to observe task T3 that is part of goal G2 (this is what they call “negative evidence”). Our framework, on the other hand, will decrease the probability of both goals due to the fading function for evidence. Even if this can be an advantage for some applications, this kind of goals which have a common prefix of actions in the plan library reduce the performance of Goldman’s et al. plan recognition system since it has to keep a large number of explanations for all the intentions that are consistent with the observed tasks. On the other hand, although both approaches allow us to model the influence of context in the user's intention, in Goldman’s et al. approach contextual information must be given beforehand are will remain fixed while in our approach they are learned from previous experiences using the user's feedback. In the same way, our approach facilitates the adaptation of probabilities in the model given the user's feedback.

The use of Bayesian inference offer many advantages for the task of inferring the user's intention, but it has also some disadvantages. First, the result of a belief update given some evidence is not sensitive to the order in which the evidence is entered in the network. In other words, if the tasks the agent observes the user performs in the application were given in a completely different order, the resulting confidence levels for non-evidenced nodes would be exactly the same. A second drawback that arises with the use of Bayesian networks is that we are
not able to handle repetitive tasks that give rise to cycles.

Finally, our framework demands a considerable effort from the designer of the agent to design the Intention Graph and the complexity of the network is increased since we add extra nodes for representing user preferences. Although the increase in the complexity limits the scalability of our approach, we believe that it is compensated by the gains in terms of early prediction of the user intentions.

References

Fig. 1. Example of an Intention Graph in the calendar management domain

Fig. 2. Example of Bayesian network built from a set of data examples
Fig. 3. Evolution of the confidence levels of each possible intention a) in the original Intention Graph and b) the Intention Graph with learnt context nodes merged into.
Fig. 4. Evolution of the confidence levels for the second scenario.

Fig. 5. Goldman, Gaib and Miller’s hierarchical plan representation in the space station domain [11]
Fig. 6. Intention Graph for the special station plan library depicted in Fig. 5

Fig. 7. Results for the special station example using Goldman, Geib and Miller’s model
Fig. 8. Results for the special station example using our approach
Personalized Detection of User Intentions

M. G. Armentano a,b,∗, A. A. Amandi a,b

a ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA, Campus Universitario, Paraje Arroyo Seco, Tandil, 7000, Argentina
b CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

Abstract

Interface agents are strategic software components for improving the quality of services to users. In order to be accepted by users, interface agents need to make useful suggestions always in the context of the user’s intention. The user’s intention should be detected as soon as possible so that the agent can define a way to collaborate with the user. Plan recognition can be applied to identify the user’s goal based on his or her actions in the environment. However, classical approaches to plan recognition fail in two main aspects that make them unsuitable for being used by interface agents: the lack of personalization and the lack of consideration of the transition between different goals pursued by the user. We propose an approach to capture intentions taking into account the variables involved in the application domain that represent the user preferences. Experimental evaluations show us that we have found a way for early detection of intentions.

Keywords: Plan Recognition, Interface Agents, User/Machine Systems, Knowledge personalization, Knowledge customization.

1. Introduction

Interface agents (Maes, 1994) are computer programs designed to assist human users in their computer-based tasks in a personalized manner. This kind of agents is able to learn the user’s interests, preferences, priorities, goals and needs.
aiming at providing him or her with proactive and reactive assistance in order to increase the user’s productivity regarding to the use of the application at issue (Schiaffino and Amandi, 2006). With the aim of assisting a user of a software application, interface agents not only have to learn the user’s preferences and habits regarding the use of the application itself, but should also consider the user’s intention before initiating an interaction with the user. Considering the status of the user’s attention (his or her intention or the goal that he or she is trying to achieve) and the uncertainty about the user’s intentions are critical factors for the effective integration of automated services with direct manipulation interfaces (Horvitz et al., 1998). As a result, we must build agents capable of detecting the user’s intention so that they can predict opportune moments for gaining the user’s attention.

Most of us have experience in some moment the situation in which a software application foretold what we were intending to do. A typical example of such application is the Office ’97’s assistant, Clippy, who interrupted users with phrases such as “It looks like you are writing a letter…” However, Clippy says the same phrase whenever the user starts writing a sentence beginning with the word “Dear …” Consequently, in these kinds of interventions, Clippy is not capturing correctly the user’s intention.

A correct detection of the user’s intention will be useful for the agent to avoid interrupting the user in an improper moment. Users generally don’t want to be interrupted while working on a specific task, unless this interruption is strongly related to the task they are performing. For this reason, most users found Clippy annoying and stopped using it (Whitworth, 2005). By considering the user’s intention the agent will be able to answer to the user’s requirements always in the context of his or her current goal. For example, if the agent observes that the user is scheduling a work meeting for the following day, the agent can offer to automatically complete the information required and to send an email to each participant of the meeting, providing that it knows the user’s preferences about the kind of meeting he or she is scheduling.

With this purpose, plan recognition aims at identifying the goal of a subject
based on his or her actions in the environment. Most of previous approaches to the problem of plan recognition fail in two main aspects for being used by interface agents. First, one of the most important problems that an interface agent has to deal with when inferring the user’s intention is the uncertainty related to the moment in which the user starts a new plan to achieve a new goal, that is how does the agent become aware that the user has already achieved one goal and started pursuing a new one? This issue is not usually addressed by many approaches to the problem of plan recognition, which consider only one "session" starting with the first observed action and ending when the algorithm recognizes the user’s intention. In an interface agent environment, the user will repeatedly start pursuing new goals in the application, with no preplanned behavior. Moreover, the user can even change his or her intention without completing his or her previous goal. This problem is usually solved restricting the memory of the plan recognizer so that it only considers the most recent tasks performed by user, or it considers each task for only a fixed interval of time and then they are completely disregarded. We present a novel alternative approach in which the strength of the evidence about the user’s actions is lessen over time.

Second, most of previous approaches do not consider adaptation to a particular user. Plan libraries are only domain dependant but user independent, because they are constructed beforehand according to dependencies in the tasks of the domain. Hand-coded plan libraries do not consider that the behavior may vary from one user to another according to the values taken by some variables of the domain. There are variables that define different contexts in which the user may have a different behavior. For example, receiving an invitation for a meeting sets a context for the user’s subsequent behavior, but if that invitation comes from his or her boss, the user may behave completely different.

To face these problems, we propose an approach that will enable an interface agent to perform personalized plan recognition over the tasks that the user executes in a software application. We will not focus on how the agent makes use of the information about the user’s intentions to provide assistance but on
providing the agent with a probabilistically ranked list of the most probable user’s intentions in every moment. The main contributions of our approach are the personalization of the plan recognition process and the use of a fading function for evidence to face the problem of transitions between goals pursued by the user. This paper is structured as follows. In Section 2 we present the problem of plan recognition. In Section 3 we present some related work. In Section 4 and 5 we present and evaluate our approach. Finally in Section 7 we present our conclusions.

2. What is plan recognition?

Consider the following situation. Mary, a secretary, observes her boss, John, opening a personal information management software application that is well known by both of them. This software application has many tools including managing events, contacts, reminders, home budget, personal diary, lists of favorite web pages, passwords, etc. Next, she observes her boss clicking on the "Add event" button. Now, she is sure he wants to add a new event to the calendar. She knows the steps required to complete this task, and she is able to help him in any moment. John enters the subject of the event, selects the day and starting time and then he clicks the "Add participants" button. After scrolling the list of contacts, he clicks on the "Cancel" button and saves the event. Mary cannot imagine what her boss is intending to do. Perhaps he did not find the contact he was looking for, or maybe he has a meeting and cannot spend more time scheduling the event. But next Mary observes that John clicks on the "Add contact" button, he completes the required information for adding a new contact and saves the changes. With the information the secretary has until this moment, she can think that the participant that the boss was intending to add to the event was not in his contact list so he had to create a new contact and next he will edit the event he had saved before to add the recently added contact as a participant of the event. Moreover, if the secretary has enough knowledge about her boss habits and the events he usually adds to the application, she could be able to predict, for example, that the event will take place in a specific
location, at a specific time just by observing the subject, the host and other information he has entered. It may also be possible for her to predict that some member of the office will take part of the meeting based on the type of meeting the boss is scheduling.

The term that has been introduced to describe the process of inferring the intentions of a subject based on the actions he or she performs in an environment is *plan recognition*. What the secretary in the previous example has done by trying to deduce her boss intention can be considered a plan recognition process. Inputs to a plan recognizer generally are (1) a set of goals the agent expects the user to carry out in the domain, (2) a set of plans describing the way in which the user can reach each goal, and (3) an action observed by the agent. The plan recognition process itself consists in foretelling the user’s goal and determining how the observed action contributes to reach this goal. The set of possible plans the user can follow to achieve a goal constitutes what is called the *plan library*.

There are currently two main approaches to the problem of plan recognition: consistency and probabilistic. Consistency approaches (Kautz, 1991; Rich et al., 2001; Gabaldon, 2009; Goultiaeva and Lespérance, 2007) aim at narrowing the set of candidate intentions by eliminating those plans that cannot be explained by the actions that the user performs. Probabilistic approaches (Horvitz et al., 1998; Brown, 1998; Huber and Simpson, 2004; Charniak and Goldman, 1991; Goldman et al., 1999; Geib and Goldman, 2005), on the other hand, explicitly represent the uncertainty associated to the user’s plans and allow to probabilistically rank the most probable user’s intentions.

Both kinds of approaches can lead to accurate predictions providing that the plan library is complete in consistency approaches, and that the probabilities are correct in probabilistic approaches. However, probabilistic approaches can find the most probable intention if the observations up to a given moment enable more than one possible intention, while consistency approaches cannot select between them and have to wait for a single consistent explanation to make a prediction. See (Armentano and Amandi, 2007) for a complete review of plan recognition for interface agents.
There are two main aspects that make classical approaches to plan recognition unsuitable for being used by interface agents. First, the agent should deal with transitions and changes in the user intentions. The agent usually will not be certain about the moment in which the user starts a new plan to achieve a new goal. Classical approaches do not consider this problem and they restrict the plan recognition process to only one "session" that starts with the first observed action and ends when the algorithm recognizes the user’s intention. In an interface agent environment the user not only will repeatedly start pursuing new goals in the application, with no preplanned behavior, but also he or she may change his or her intention without completing the previous one. Those approaches which do consider this problem restrict the memory of the plan recognizer so that it only considers the most recently performed tasks, or they consider each task only for a fixed interval of time, completely disregarding them after that “lifetime”.

Second, user preferences have a fundamental role for an interface agent and should be considered in the plan recognition process. Considering the user’s preferences in the plan recognition process is an important factor that an interface agent should regard because the behavior of the user in a specific situation is usually determined by his or her preferences related to that situation. In this context, a situation is described by the information the user is handling to achieve his or her intention. In the calendar scheduling domain, for example, the situation related to a new event addition may be described by the event’s date, time, participants, place, subject, etc. These variables of the domain strongly influence the user behavior. For instance, if the user adds a new event which time is within his or her working hours, there is a high probability of the event taking place in the office.

The process of adapting a computer application to the needs of a specific user is called personalization and takes advantage of knowledge acquired from the analysis of the user’s behavior and data. In contrast to customization which is a user-initiated and user-driven process, personalization is system-initiated and system-driven and thus requires the system to monitor the user’s behavior.
in order to adapt automatically. The fundamental purpose of personalization is
the user’s satisfaction, and it is motivated by the acknowledgment that a user
has needs and preferences, and meeting them successfully is likely to lead to a
satisfying relationship with him or her.

The importance of personalizing the plan recognition process by considering
user preferences has been identified previously in (Bauer, 1998) where decision
trees where used to learn classes of situations based on the user’s actions. This
approach runs the plan recognizer on a set of input examples from a typical
user, and then gathers statistical data based on the results of running the plan
recognizer on the entire observable behavior in each episode. The main problem
of using decision trees to "classify" the intention of the user according to some
attribute of his or her actions is that they can only assign one possible intention
given those attributes, completely discarding other options.

In the next section we describe some related work in the area of plan recog-
nition. Then, in Section 4, we present our approach to provide an interface
agent with a probabilistic ranking of the most probable user’s intentions which
takes into account the user’s preferences.

3. Related work

The prediction of the user’s goal is an inherently uncertain task. There-
fore, it is desirable to have a knowledge representation capable of dynami-
cally capturing and modeling the uncertainty of human-computer interactions.
Non-probabilistic approaches (Kautz, 1991; Rich et al., 2001; Goultiaeva and
Lespérance, 2007) are weakened by the fact that they cannot decide to what
degree the evidence supports any particular hypothesis about the user’s goal.
This is an important issue to consider for the agent to be able to rank differ-
ent possible explanations supported by the set of performed actions. Markov
models and Bayesian networks are two of the most important representations
for dealing with this kind of information.

Mott et al. (Mott et al., 2006) performed a set of empirical studies on two
families of probabilistic goal recognizers, n-gram models and Bayesian networks
and conclude that probabilistic approaches can perform goal recognition that is accurate and incrementally converging.

The main advantage of n-gram models (Markov chains) is that they can be derived from data without any previously known structure. However, the Markov assumption is unrealistic in our domain. A major disadvantage of higher order Markov chains (n-grams, for n > 1) is the exponential growth in the state space when we increase the order of the model to capture more than very short term memory dependencies in the sequences. Blaylock and Allen (Blaylock and Allen, 2003) explored the use of n-gram models to perform statistical goal recognition. Specifically, they built two experiments in the Linux domain using unigram and bigram models, respectively. Then they defined the problem of goal recognition as a classification task where given an observed sequence of k instantiated actions observed until a given moment, the algorithm searches for the most likely instantiated goal.

Bayesian networks have also been successfully used in previous approaches to plan recognition. Charniak and Goldman (Charniak and Goldman, 1991) built the first probabilistic model of plan recognition. They use a quantifier-free first order language as representation and belief nets for plan inference. The random variables in the belief net are propositions, whereas the root nodes are hypotheses about agent’s plan. Charniak and Goldman apply their plan recognition system in the area of story understanding. They apply a set of rules to a sequence of observations using plan knowledge, to automatically generate a Bayesian Network. Then, after incorporating prior probabilities, this network supports the selection of possible explanations of observed actions using the posterior probability of each hypothesis, computed by propagating the values from the evidence in the net. This approach assumes that the plan library is represented in a quantifier-free first order language. The network is built from rules over that representation, but it is not adapted to a particular user.

Brown (Brown, 1998) also used Bayesian networks to build a user profile that enables the detection of a user’s intentions considering the utility of offering assistance. This approach is based on the belief that what the user is
intending to do in an environment is the result of the stimulus (i.e. the events) that occurs in the environment, and of the goals he is trying to reach as a reaction to those stimuli. Goals are made of a set of actions, with pre and post conditions. Preconditions include both events that are directly observable in the environment (such as the user clicking in a button labeled “Add event”) and events that are indirectly observable (such as increase cognitive load of the user). These preconditions cause the user to intend to achieve a goal. Goal abstractions facilitate the design and detection of higher level goals in the search of lower level goals. Evidence can be added and removed easily and intuitively (in the form of pre and post conditions) as the user interacts with the system. Pre and post conditions for goals and actions can be explicitly indicated, and keyhole plan recognition is made easier by enumerating explicitly the actions that make up goals. Brown assumes that the user is never observed to perform a goal explicitly. Although this assumption is correct, the user’s feedback can be considered by the agent to know in a given moment the user intention.

More recently, Goldman et. al. (Goldman et al., 1999; Geib and Goldman, 2005) presented a probabilistic theory of plan recognition based on plan execution. They use a simple hierarchical (task decomposition) representation of plans that provides recipes for achieving goals. In their model of plan execution, initially the user has a set of goals and chooses a set of plans to execute to achieve those goals. The set of plans chosen determines the set of pending primitive actions. Then the user will repeatedly execute one of the pending actions and generate a new set of pending actions from which further action will be chosen. The new pending set is generated from the previous set by removing the last executed action and adding the actions that become enabled by it execution. Actions become enabled when their required predecessors are completed. The algorithm first computes the complete and covering set of possible explanations; second, it computes the probability of each of the explanations; finally, it computes the conditional probability of the given goal on the basis of the probability of the explanations. A disadvantage of this approach is that the number of explanations can grow exponentially in the number of root goals
that share a common unordered prefix of actions.

When a plan recognizer is to be adapted to a particular user of a software application, a proper model of his preferences is crucial because preferences are the reasons why a user acts in a particular way when facing a certain situation (Bauer, 1998). For example, a user of a scheduling application can prefer organizing events in his house when the participants are his or her friends or when it takes place in the afternoon, but might prefer the club for events which take place in the morning. Ignoring the user’s preferences hinders the plan recognition process, especially if the behavior of the user deviates from what is typically to be expected in a domain.

Machine learning techniques are the most widely used method to build user models implicitly. The main reason of the popularity of these techniques is that user models can be obtained incrementally, usually without intervention of the user, by observing the user while he performs his tasks.

Besides Bauer’s approach described in Section 2, other approaches have been proposed for acquiring domain knowledge by probabilistic reasoning. Chen et al (Chen et al., 2002), for example, presented a method to model and inferring user’s intention using data mining. They make the distinction between action intention (low level, such as mouse click, keyboard typing and other basic actions performed on a computer) and semantic intention (what the user wants to achieve at high level, which may involve several basic actions on a computer to accomplish it). They work in the domain of a web browser, and mainly focus on predicting action intention based on the features they extracted from the user interaction such as user’s typed sentences and viewed content. A modified Naïve Bayes classifier is used to model the user’s action intention on a computer. The model can be trained incrementally and used to predict the user’s next action. However, they only focus on the prediction of action intentions, i.e. the prediction of the next action of the user based on the document he or she is browsing or what he had typed in a web browser. Plan structure is not considered in this approach.

Maragoudakis et al. work (Maragoudakis et al., 2003) is based on inferring
intentions for actions from questions in an educational dialog engine. They use automatic encoding of the semantic model of the application, based on learning Bayesian networks from past user questions. Both the structure of the networks and the conditional probability distributions are computed automatically from dialog corpora. This framework is Bayesian in that they start from a causal theory of how the agent’s mental state causes its plan, executing its plan causes activity, and they reason from observed effects to underlying causes.

Nkambou et al. (Nkambou et al., 2011) proposed a procedural-knowledge acquisition framework based on a combination of sequential pattern mining and association rules discovery techniques. This framework aims at providing Intelligent Tutoring Systems with relevant domain knowledge that enable them to guide the learner during problem-solving learning activities for ill-defined domains where this knowledge is hard to be defined explicitly by domain experts. The framework builds a meta-knowledge base of plans from users’ traces that is then used to better help the learner on demand.

In the next Section we present our approach for providing an interface agent with a probabilistic ranking of the most probable user’s intention.

4. Probabilistic ranking of user intentions

In general, a user performs a set of tasks in order to achieve a higher level goal. For example, to arrange a dinner with his or her friends, the user has to look at the calendar for a convenient date, he or she has to create a new event in his or her agenda and has to write an invitation mail to each participant of the meeting. Therefore, when the user performs a task, the agent should consider the set of possible intentions associated to it to find a way to collaborate with the user. The prediction of the user’s intention is an inherently uncertain task. Therefore, we look for a knowledge representation capable of capturing and modeling dynamically the uncertainty of human-computer interactions.

In our approach, we represent the set of intentions the user can have in the application domain with an Intention Graph. We define an Intention Graph to be a representation of the context of execution of tasks and it is materialized
by a Bayesian network. The context is viewed as the set of tasks that the user has performed recently, and will influence the confidence that the agent has in any given intention that the user may have.

Bayesian Networks are a probabilistic knowledge representation used to represent uncertain information. Bayesian networks are directed acyclic graphs representing both the conditional dependencies and independencies between elements of the domain. Knowledge is represented by nodes called random variables and arcs representing causal relationships between variables. Each variable has a finite set of mutually exclusive states. Nodes without a parent node have an associated prior probability table. On the other hand, the strengths of the relationships are described using parameters encoded in conditional probability tables (CPTs).

Bayesian networks are used for calculating new probabilities when some particular information becomes available. The information is called evidence and will be in the form $A = a$, where $A$ is a variable and $a$ is a state of $A$. In our case, we will have new evidence every time the user performs a task in the software application. Therefore, evidence will be in the form “AddContact=true” meaning that the user performed “AddContact” task.

The general setting for probabilistic inference is that we have a set $V = V_1, V_2, ..., V_k$ of propositional variables and we are given as evidence that the variables in a subset $E \subseteq V$ have certain definite values, $E = e$. In the probabilistic context, we want to compute the conditional probability distribution $p(V_i|E)$, for each variable $V_i$ given the evidence. This process is called probabilistic inference.

By making use of probabilistic inference and considering the set of tasks performed by the user as evidence, we will be able to compute the probability that the user is pursuing any given intention modeled in the Intention Graph. Moreover, if the user explicitly declares his or her intentions, we will be able to probabilistically assess the tasks that he or she has to perform to achieve his or her goal.
4.1. Building an Intention Graph

In our Intention Graph variables correspond to goals that the user can pursue in the application domain and to tasks that the user can perform in the application to achieve those goals. The two possible states of these variables are true and false. A true value in a variable representing a goal indicates that the user is pursuing that goal. On the other hand, a true value in a variable representing a task indicates that the user has performed that task in the application. We call confidence level to the probability of a variable being in state true. Notice that we will not have direct evidence about the goal the user is pursuing unless the user makes an explicit declaration of his or her intention. Evidence on a task node will be set when the user interacts with a widget in the application GUI that is associated to the execution of the corresponding task. Finally, our Intention Graph includes a third kind of variable: context variables. This kind of variables will be used to personalize the intention detection process and will be explained later in Section 4.4.2

In order to construct the graph, an expert in the application domain should first add a node for each task and intention in the domain and then draw arcs according to existing relationships among tasks and intentions. Missing arcs encode probability conditional independence assertions, as no dependency exists among unrelated nodes. Considering that the tasks that the user performs are caused by what he intends to do in the application, the direction of arcs in our Intention Graph is always from goals to tasks. Relations among tasks are also allowed and indicate that the execution of one task influence the execution of other task.

Once the structure of the Intention Graph is built, the domain expert should assess the strength of the relations between variables my means of conditional probabilities tables in each node with parents in the graph and a priori probabilities for nodes without parents. Although the probabilities selected will have a strong influence in the quality of the Intention Graph, they can be automatically updated later by using the user feedback about his or her intentions, as we will detail in Section 4.4.
4.2. Example

In order to illustrate our proposed approach we will use an application called Agenda 4.0 that provides various functions for managing personal information such as contacts, tasks, events, and email.

The calendar domain is a well-known domain in which interface agents can be applied. An interface agent in this domain can assist users with tasks such as suggesting meeting places, dates and times for a new event the user is scheduling, suggesting the participants for a meeting, notifying the user when an event is about to happen and warning about overlapping events and about an unusual event scheduled by the user. As stated before, to provide this functionality, the agent should be aware of the set of tasks that the user is performing because it gives the context in which the user is putting his or her attention. This context will enable the agent to infer the user’s intention and it will be able to provide assistance to the user, or it can collaborate with the user to fulfill the intention he or she has.

The Intention Graph we designed for this application is shown in Fig.1. Nodes labeled with black background and white text correspond to intention nodes and nodes labeled with white background and black text correspond to task nodes.

As we can see, we identified six intention nodes: "InviteToMeeting", "SendMailToContact", "RegisterBirthday", "ContactBirthday", "EventBirthday" and "AddMissingContactGroup". "InviteToMeeting" reflects the intention of the
user of arranging a meeting with a selected contact. On the other hand, "SendMailToContact" reflects the intention of the user of sending an email to the contact he has selected in the application. "RegisterBirthday" is a goal that can be achieve either by registering the birthday of a selected contact in the personal information of that contact ("ContactBirthday" intention) or by creating a new event to remember about that important date ("EventBirthday"). Finally, "AddMissingContactGroup" is used to detect when the user has to add a new contact group to the agenda to be used afterwards as the group of a given contact. Although a simple example, in Section 5 we show how beliefs on the intentions modeled in the Intention Graph are updated as a user interacts with the application.

4.3. Fading Past Observations

Recall the example of the secretary presented in Section 2. What should Mary, the secretary, think if before completing the intention she believes her boss has, John receives a telephone call and Mary observes that he makes a new entry in his personal diary in the application? Should she consider that her belief was wrong? John can yet continue editing the event after saving the new entry in the diary. Or perhaps John has abandoned his previous goal and started a brand new one. This example shows that plan recognizers must be careful not to over-commit themselves to one possible intention. Especially in a domain where users can change they current goal rapidly, it is often a good idea for the plan recognition system to be a little "forgetful".

As stated in Section 2, most of previous approaches to the problem of plan recognition do not consider the uncertainty related to the moment in which the user starts a new plan to achieve a new goal. Those which consider this issue limit the memory of the plan recognizer by making evidence to be present in a fixed interval of time and then completely disregarding it. We take a different approach in which evidence is gradually forgotten.

In Bayesian networks terms, evidence is a collection of findings on variables. A finding may be hard or soft. A hard finding specifies which value has taken a
given variable. Findings on the values taken by the variables introduced before are of this kind. A soft finding, on the other hand, specifies the probability distribution over a variable’s possible values. Hard evidence is a collection of hard findings and soft evidence is a collection of soft findings.

In our approach, we adopt the concept of soft evidence to fade the evidence we entered to the Bayesian network as the user performs further tasks. To do so, we use a fading function \( F(\cdot) \) to gradually forget the tasks performed by the user, as shown in Eq. 1.

\[
P(t = true) = P_{prev}(t = true) - F(\cdot)
\]  

That is, the current probability that task \( t \) was performed, \( P(t = true) \), is computed by subtracting from the previous probability assigned to task \( t \), \( P_{prev}(t = true) \), a value assigned by the fading function \( F(\cdot) \). Evidence on every node is faded according to this function until they reach their original value, i.e. until the confidence of a given node is lower than the value that it would have if we wouldn’t have observed the execution of the corresponding task in the application.

By gradually forgetting past observations, the agent not only will be able to manage changes in the user’s goal, but also will let it to forget the execution of noisy tasks, that are tasks that do not belong to the main goal the user has in a given moment. In our example, the boss can check the current date for its own sake, and not because it is part of the plan he is following in that moment.

Fading functions can be any function that, given the Intention Graph and the evidence on tasks performed so far, gradually decrements the certainty of the evidence according to some heuristic (Liu et al., 2007). For example, we can decrement current evidence by a fixed factor \( 0 \leq \triangle \leq 1 \) every time the user performs a task in the application. This way, for all nodes with evidence \( t_i \) we will update the probability distribution of its evidence according to Eq. 2

\[
P(t_i = true) = P_{prev}(t_i = true) - \triangle
\]
This simple fading function will allow the agent to disregard, after some actions performed by the user, a previously performed noisy action or a previously pursued goal. This function, however, might have the problem of rapidly forgetting evidence in tasks that actually contribute to the user’s current goal. This problem can be solved using another function that, for example, keeps almost intact the evidence of some number of tasks and then quickly decrement up to the original value of the node (without evidence).

4.4. Personalizing the detection of the user’s intentions

When a plan recognizer is to be adapted to a particular user of a software application, a proper model of his or her preferences is crucial because preferences influence the way that a user acts when facing certain situations (Bauer, 1998). In our example, the secretary should learn that her boss prefers organizing events in his house when the participants are his friends or when it takes place in the afternoon but that he prefers the club as the meeting place for meetings taking place in the morning. Ignoring the user’s preferences hinders the plan recognition process, especially if the user’s behavior deviates from what is typically expected. So, an agent provided with plan recognition and user modeling capabilities can detect patterns of behavior that are particular to each different user of the application.

In our approach, we personalize the Intention Graph initially constructed by the domain expert in two ways: (1) by adapting the probabilities given new user experiences (Section 4.4) and (2) by learning new relations between the attributes of the tasks that influence the user’s intentions (Section 4.4.2).

4.4.1. Adaptation of probabilities in the Intention Graph

As stated above, by using Bayesian networks inference mechanism the agent will be able to update its belief in any given goal related to the tasks performed by the user. This belief update is made using the probabilities set by the domain expert who designed the Intention Graph.

However, when constructing a Bayesian network the domain expert may be uncertain of the correctness of the conditional probabilities chosen. Neverthe-
less, when the agent is at work it will repeatedly get new experiences of tasks performed by the user when he or she has some particular goal. It would be desirable that the agent could learn from these new cases to adapt the probabilities set by the domain expert who has designed the Intention Graph. This adaptation process will make it possible to model the way of acting of a particular user.

For this task we take a statistical on-line learning approach. While the user interacts with the application, the agent “remembers” the tasks performed by the user. Then, when the agent infers that the user is pursuing certain goal, it adapts the probabilities of the intention graph to reflect this new experience. The agent will believe that the user has already completed the current goal(s) when a certain threshold is exceeded. This threshold should be empirically determined for each application domain. Explicit feedback can also be considered if the agent believes that it can ask the user whether he is trying to achieve a given goal or not, but this decision is out of the scope of our work.

The statistical method we used to update the CPTs of the variables in the network is called Fractional Updating and was proposed by Jensen (Jensen, 2001). The statistical task is to gradually modify the estimates of the parameters with the new experiences entered. To do so, besides the CPT for each node, we keep an experience table. The experience table of a variable represents the experience counts of the parent configurations. For example, in the Intention Graph in Fig.1, the experience table for the node labeled “Events” will have two entries, one indicating the number of times we observed the execution of “Events” task when the user intended to register an event to remember about a birthday of a specific contact (“Event-Birthday” intention) and the other indicating the number of times we observed the execution of “Events” task when the user was not pursuing “Event-Birthday” intention.

Let $n_k = P(N = k)$, the probability of node $N$ being in state $k$, $y_k = P(N = k|e)$ the probability of node $N$ being in state $k$ given the current evidence in the Bayesian network, $conf$ a given configuration of the parent nodes of $N$, $z$ the probability of such configuration given the current evidence and $n_{prev}$ the
previous experiences count for $n_k$. We update the CPT for node $N$ being in its state $k$ for a given parent configuration $conf$, according to Eq. 3.

$$P(n = k|conf) = \frac{n_k + z \cdot y_k}{n_{prev} + z}$$ (3)

For further information about Fractional Updating, please refer to (Jensen, 2001).

4.4.2. Learning new relations in the Intention Graph

A further adaptation that can be done to the Intention Graph built by the domain expertis to learn new relations that may arise between the attributes of the tasks performed by the user and the intention nodes in the Intention Graph.

For example, in Agenda 4.0, the user can select a contact from the address book with the objective of sending this contact a mail or with the objective of scheduling a meeting with this contact, as shown in the Intention Graph presented in Fig.1. The Intention Graph constructed manually by a domain expert will enable the agent to rank which of these two goals is more probable, given that the user has selected a contact from his or her address book. However, the information about the selected contact can be relevant in discerning which goal the user actually has.

To consider this information, we introduce to the definition of our Intention Graph, the concept of *traceable nodes*. A traceable node is a node of the Intention Graph in which we want to register the values taken by some attributes of the corresponding task when it is performed by the user with. These attributes will be modeled as new variables that represent the context in which the user performed the task corresponding to a traceable node and will be used to find new relations between these variables and the nodes in the Intention Graph.

Every time the user performs a task corresponding to a traceable node, the agent will observe the values taken by the attributes of the task (for example, the selected contact is from New York and belongs to his group of friends). Then, the agent will continue observing the user until it can infer which his or her intentions are and will save the experience in an interaction history. Each experience will be
Figure 2: Example of Bayesian network built from a set of data examples

in the form $<\text{attribute}_1, \text{attribute}_2, ..., \text{attribute}_n, \text{intention}_1, ..., \text{intention}_k>$

where $\text{attribute}_i$ is the value taken by the attribute $i$ and $\text{intention}_j$ is true if the agent infers that the user was pursuing intention $j$ or false otherwise.

This database of experiences is then used by the agent to run both a batch learning and a parametric learning algorithm to find relations between the attributes themselves and between the attributes and the intentions. In our implementation, we use the PC algorithm (Spirtes et al., 1993) for learning the structure of the network from the database of experiences and the EM algorithm (Lauritzen, 1995) for learning the probabilities in the CPTs. Fig.2 shows an example of a network learnt from experiences in the scheduling application.

In this machine-learned Bayesian network, variables representing attributes are incorporated to the network in the form of a new kind of variables, context nodes (gray nodes correspond to this kind of nodes nodes), and variables representing intentions as intention nodes. The learnt network is then merged with the Intention Graph to incorporate this knowledge to the detection of the user’s intention.
5. Experimental results

To perform our experiments, we selected the calendar management domain for two reasons. First, because it is a well known and easy to communicate domain in which interface agents can be applied. Second, because of the richness in the number of variables involved which make it a non-trivial domain. An interface agent in this domain can assist users with tasks such as suggesting meeting places, dates and times for a new event the user is scheduling, suggesting the participants of a meeting, notifying the user when an event is about to happen and warning about overlapping events and about an unusual event scheduled by the user. A complete development of such an agent is out of the scope of this work. We are concerned only in providing an ordered ranking of the possible goals the user can pursue.

In this section, we present two scenarios. The first scenario aims at evaluating the influence that contextual information representing the user's preferences has in the prediction of the user's intentions. The second scenario shows how the Intention Graph, due to the fading function, reacts when the user performs noisy tasks.

5.1. Scenario 1

With the objective of showing the influence of the user's preferences in the detection of the user's intention, we select a scenario observed from the interaction of a regular user of the scheduling application in which he utilized the scheduling application to organize a meeting with some contact in his address book and then selected another contact to register his or her birthday. To achieve these goals, the user performed the following sequence of tasks: <"SelectContact", "AddContactToMeeting", "SelectContact", "EditContact", "PersonalInformation", "EnterBirthday">.

We recorded the confidence levels for all intentions both in the Intention Graph containing the user’s preferences information and in the same Intention Graph without context nodes (the one owned originally by the agent). Fig.3.a shows the evolution of the confidence levels of each possible intention in the
original Intention Graph without context nodes. In the first time slice, we show the a priori probabilities of each intention when the user did not perform any tasks in the application. "SendMailToContact" is the more probable intention, while "ContactBirthday" is the least probable one. When the user performed the first task, "SelectContact", the ranking remained unchanged, although there was a little increment in those intentions that contained this task. Then the user performed "AddContactToMeeting" and "InviteContactToMeeting" became the most probable intention. With the following set of tasks performed by the user, "ContactBirthday", "RegisterBirthday" and "EventBirthday" started gaining confidence while the other intentions decreased their confidence level. "AddMissingContactGroup" confidence level remained unchanged along this session because it is "disconnected" from the tasks performed by the user. The agent in this case considered a threshold level with a value of 0.7 to believe in the intention pursued by the user; it predicted the first intention after observing the second performed task and the second intention after observing the fifth task. If we consider the number of tasks in each intention, it needed both tasks to be performed to detect "InviteToMeeting" intention, and three tasks out of four to detect "ContactBirthday" intention.

Fig.3.b shows the same scenario but performed using the Intention Graph with context nodes merged. The first "SelectContactTask" was performed when the user selected a contact from the "Friends" group, living in "New York", and that already has the birthday registered. We can see that the confidence level for "InviteToMeeting" is higher only with the first task performed. We can also see that the confidence levels of the remaining intentions were dramatically lowered. It is logic to think that the user would not register a birthday in the contact information because the selected contact already had a birthday date set in the address book. The second contact selected by the user was also from the "Friends" group and its city was "New York", but the birthday was not set yet in this case. So, we can see that with the mere selection of the contact, "ContactBirthday" intention could be predicted.

A remarkable point in the experiment performed is that the curves corre-
Figure 3: Evolution of the confidence levels of each possible intention a) in the original Intention Graph and b) the Intention Graph with learnt context nodes merged into
sponding to the possible intentions of the user are closer when we do not consider the user’s preferences than when we consider them. This fact indicates that although one intention can be more probable than others, the confidence levels of the other intentions have a similar (low) value. Therefore, the incorporation of the user’s preferences enables a better distinction of the actual user intention.

Another interesting fact that can be appreciated in Fig. 3 is that the confidence of finished intentions gradually decrements to its original value, as happens with "InviteToMeeting" intention. This is due to the fading function used by the intention graph that gradually decrements by a fixed constant the strength of the evidence on the performed tasks.

5.2. Scenario 2

In this second scenario we test how the Intention Graph reacts when the user performs a “noisy” task and follows two intentions at the same time. In this scenario, the user forgot greeting one of his friends living in Buenos Aires about his birthday, so he intends to create a reminder in the application for the next year, register the birthday date in the personal information of his friend, and write an email to congratulate him and to apology for his absentmindedness. The ordered set of tasks performed in this scenario is <"Events”, “AddContact-Group”, “New Event”, “SelectContact”, “EditContact”, “PersonalInformation”, “EnterBirthday”, “SendMail”>. The intentions followed by the user that lead him to perform those tasks are “EventBirthday” which is interrupted by the execution of “AddContactGroup”, “ContactBirthday” and “SendMailToContact”.

Fig. 4 shows the evolution of the confidence level for the intentions in the Intention Graph as the user performs the mentioned tasks. When the user changes to the events tab in the application, the confidence for registering a birthday by creating an event increases near to 0.78. Then the user performs a noisy task and the confidence level of this intention decreases, but when the user resumes the previous intention by performing “NewEvent” it is increased again. However, we should notice that the increment in the confidence after resuming the intention that was interrupted by the noisy task is not very significant. This
is due to the fact that the noisy task caused a decrement in the strength of the evidence we had on the “Events” task.

The selected contact in the fourth task was part of the “Friends” group, lived in “Buenos Aires”, was a “System Engineer” and did not have a birthday registered in the application. When the user selected this contact for edition, “ContactBirthday” becomes the most probable intention. However, due to the characteristics of the selected contact, “SendMailToContact” also increases its confidence level. As the user continues performing tasks in the application to achieve his intention of registering the birthday of the selected contact, the confidence level on “SendMailToContact” is gradually decreased. However, when the user finally performs “SendMail” task, it gets a value above 0.9.

6. Comparison with Goldman’s et al. approach

Our model is situated within probabilistic approaches to the problem of detecting the user’s intentions and combines both keyhole and intended plan recognition. Keyhole plan recognition is accomplished by observing the actions
performed by the user in a non-intrusive manner. Intended plan recognition is incorporated to the model by allowing the user to provide feedback about his intentions. This explicit feedback will allow the agent to adapt the intention model to a particular user of the application.

From the plan recognition approaches presented in Section 3, the one that is most similar in spirit to our model is the model for plan recognition proposed by Goldman et al. (Goldman et al., 1999; Geib and Goldman, 2005). This approach handles partially-ordered plans, interleaved plans, considers the effect of context in plan choice and is able to manage the execution of actions for its own sake. It also supports missing observations, and noisy tasks that do not contribute to the main goal of the user.

Their approach is both consistency and probabilistic and is based on plan execution. Plans are represented as a hierarchy (task decomposition) providing recipes for achieving goals. Fig. 5 shows an example plan library for a hypothetical space station example, as presented in (Goldman et al., 1999). Tied arcs from a parent node to a child node represent “and” nodes, while untied child represent “or” nodes, or alternative plans to achieve the (sub) goal represented by a node. For example, it is possible to increase the power (“increase-power”) either by generating power (“gen-power”) or by using less power (“lower-power-use”); to generate power, it is need to open panel 1 (“open-p1”) and to start generator B (“start-gen-B”). Horizontal arrows represent temporal restrictions. For example, before starting generator B, it is needed to open panel 1. Dashed lines represent the influence of context on the possible goals. For example, the existence of an oxygen drop (“o2-drop”) influences the fact that the user will intend to raise the level of oxygen (“raise-o2-level”).

In Goldman’s et al. model of plan execution, the user initially has a set of goals and chooses a set of plans to execute to achieve those goals. The set of plans chosen determines the set of pending primitive actions. Then the user will repeatedly execute one of the pending actions and generate a new set of pending actions from which further action will be chosen. The new pending set is generated from the previous set by removing the last executed action.
and adding the actions that become enabled by it execution. Actions become enabled when their required predecessors are completed.

They define an explanation of a set of observations as a minimal forest of instances of plan trees with expansions chosen for “or” nodes sufficient to allow an assignment for each observation to a specific basic action in the plan. The algorithm first computes the complete and covering set of possible explanations; second, it computes the probability of each explanation; finally, it computes the conditional probability of the given goal on the basis of the probability of the explanations.

In order to make Goldman’s et al. model probabilistic, it is necessary to specify probabilities for a restricted set of hypothesis. First, we have to specify the probabilities for the top-level goals which may be a priori, or conditioned on the context. In the example in Fig. 5, we have to specify probabilities for “increase-power”, “raise-o2-level” and “raise-temp”. Then we have to set the probabilities that the user will chose a particular method when attempting to achieve a goal. In the example, the probability that the user will choose to perform “gen-power” or “lower-power-use” when he aims at increasing the power, and “lower-o2-use” or “gen-o2” when he intends to achieve “raise-o2-level”. Finally, we have to detail the probabilities that the user will chose a particular primitive action from a set of pending primitive actions.
Based on these probabilities, the probability of a given explanation is computed by multiplying together the priors for each goal, the probability of the expansion choices at each “or node”, and the probability of the observed actions being chosen from the pending sets:

\[
p(\text{explanation}|\text{observations}) = \prod_{i=0}^{I} p(\text{goal}_i) \prod_{j=0}^{J} \frac{1}{|\text{Choice}_j|} \prod_{k=0}^{K} \frac{1}{|P\text{S}_k|}
\]

where the first term is the probability of the user’s goal, the second term the probability that the user chooses the particular plans to achieve those goals, and the final term captures the probability that the given observations occurred in the specified order. In the formula presented above, it is assumed that each method is equally probable given its parent goal and that actions are equally likely to be chosen from the pending set.

Having the probability of each explanation given the observations, the conditional probability of a specific goal \(g\) is computed by summing the probability mass associated with explanations that have the goal present and dividing it by the total probability mass of all the explanations:

\[
P(g|\text{observations}) = \frac{\sum_{e}^{Exp} p(e, \text{observations})}{\sum_{e}^{Exp} p(e, \text{observations})}
\]

where the denominator sums the probability of all explanations for the observations, and the numerator sums the probability of the explanations in which goal \(g\) occurs.

6.1. Experiment configuration

In order to compare the behavior of our model with Goldman’s et al. model we will use the sample plan library for the hypothetical space station example in Fig. 5. We will not consider the “context” nodes (“EVA-prep” and “o2-drop”) for simplicity. The Intention Graph we build for this plan library is shown in Fig. 6.

For both models we performed the following sequence of tasks: <“open-p1”, “check-temp”, “start-gen-B”, “raise-temp-set”, “open-p1”, “open-p2”, “shutoff-x2”, “start-gen-B”, “start-o2-gen”, “check-temp”, “shutoff-x1”>. This complex
task sequence contains multiple interleaved goals and noisy tasks. With the first four tasks, the user’s goals are both “increase-power” and “raise-temp” goals. Then the user intends to achieve both “increase-power” and “raise-o2-level” and he or she performs a noisy “check-temp” in the way of achieving those goals. We analyze the behavior of the probabilities for the top level goals. Fig. 7 shows the results obtained for Goldman, Geib and Miller model and Fig. 8 shows the results obtained with our approach.

We are interested in comparing the behavior of the curves obtained for each intention, rather than the probability values obtained for each intention at each time step. The first difference to notice is that in Goldman’s et al. approach, the probability values for all intentions before any task is performed is 0 while our model is able to predict each goal with an a priori value. This reflects the fact that when no evidence is observed the user is more likely willing to achieve some goals rather than other goals. When the user performs “open-p1”, probabilities for both “raise-o2-level” and “increase-power” grow in both approaches. However, in Goldman’s et al. approach, this probabilities remains constant until the user performs “raise-temp-set”. This is due to the fact that, with the evidence available until this moment, the model cannot decide which of the top level intentions the user has since both “open-p1” and “start-gen-B” are part of both candidate intentions. In our model, on the other hand, “increase-
Figure 7: Results for the special station example using Goldman’s et al. model

Figure 8: Results for the special station example using our approach
"power" is always more probable due to the a priori probabilities encoded in the model. The variations in the values of these intentions are due to the fading function applied in our model. When the user performs "raise-temp-set", in Goldman’s et al. approach the system starts to believe that "increase-power" is more probable than "raise-o2-level" because it fails to observe "start-gen-B". On the other hand, in our model the probabilities for both intentions decrease because of the fading function.

When the user performs "open-p1" again, both intentions ("raise-o2-level" and "increase-power") gain confidence in both models. In our model, this reflects the fact that evidence on "open-p1" is reinforced. On the other hand, in Goldman’s et al. approach the increment in the confidence in both intentions occurs because they create a new set of explanations containing new instances of plans containing this task.

When "open-p2" is performed both models considerably increase their belief in "increase-power", since "open-p2" is a task that only contributes to "increase-power". However, in this situation there is an important difference between both approaches. Goldman’s et al. approach will maintain almost constant its belief in "increase-power" until the user performs the last task ("shutoff-X1"). Instead our model gradually decrements its belief in "increase-power" since the user continues performing tasks that contribute to other intentions and will increment its belief in "increase-power" again when the user finally performs "shutoff-X1".

In the example, "check-temp" was performed as a noisy task. With the observation of this task, Goldman’s et al. model will believe the user is pursuing "raise-temp" intention and will remain expecting the user to perform "raise-temp-set", a fact that will not occur. Our approach does not have this problem since evidence in "check-temp" will be gradually forgotten by the model due to the fading function.
6.2. Discussion

Both approaches are able to handle partially-ordered plans, interleaved plans and consider the effect of context in plan choice. One advantage of Goldman’s et al. model with respect to our model is that it supports “negative evidence”: the confirming or disconfirming effect of failing to see some action. In the scenario presented above, when the user performs “raise-temp-set”, it fails to observe the execution of “start-o2-gen” that would contribute to “raise-o2-level” goal, so it decrements its belief in this intention and increments it belief in “increase-power” which contains “open-p1” and “start-gen-B” but not “start-o2-gen”.

Although not shown in the example presented above, both approaches allow to model the influence of context in the intention the user is pursuing. However, in Goldman’s et al. approach contextual relations are fixed while in our approach they are learned from previous experiences. In the same way, our model enables adaptation of probabilities given the user’s feedback.

An important disadvantage of Goldman’s et al. approach is that the number of explanations can grow exponentially in the number of root goals that share a common unordered prefix of actions. These “shared leaders” appear in domains where many different plans for different goals have a common prefix of actions, and have a significant impact on the algorithm’s run-time. Plan libraries without ordering constraints represent the worst case for the algorithm performance. In our proposal, there is also an increment in the run-time when we add context nodes representing the user’s preferences. However, this increment in run-time is compensated by the fact that the user’s intentions are detected earlier using this contextual information.

7. Conclusions

We presented a probabilistic approach that can be used to enhance an interface agent with the capability of detecting the user’s intentions. This approach enables a probabilistic ranking of the possible intentions that the user may have based on the evidence of the tasks performed by the user.
Furthermore, our approach combines information about general intentions that any user can achieve in the application with specific information of each particular user regarding those intentions. This is a necessary issue to consider when designing a plan recognizer that aims at being used by an interface agents and that is often not addressed by general approaches to the problem of plan recognition.

We performed a set of experiments to test the ability of our Intention Graph to rank the user’s intentions based on the evidence about the performed tasks. With these experiments we conclude that the user’s preferences have a fundamental role in the plan recognition process because they enable a better distinction of the user’s actual intention. On the other hand, the fading function in the Intention Graph enables a gradual "oblivion" of the evidence that enables our approach to adapt to changes in the user’s intentions.

We compared the behavior of our framework to that proposed by Goldman’s et al. This model has a different behavior compared to our model when facing a set of tasks that can be performed to achieve more than two goals. For example, suppose that goal G1 is achieved by performing tasks T1 and T2 and goal G2 is achieved by performing tasks T1, T2 and T3 and the system observes tasks T1 and T2. By using the model proposed by Goldman’s et al. we are not able to predict which goal the user is pursuing because we have two explanations for the performed tasks so each one will have the same probability of 0.5. Our model, on the other hand will rank goals G1 and G2 according to their prior probabilities and the propagation of the evidence introduced in the Bayesian network. If task T4 is observed next, we can remark another difference between both approaches. Goldman’s et al. model will increase the probability of goal G1 as it failed to observe task T3 that is part of goal G2 (this is what they call “negative evidence”). Our framework, on the other hand, will decrease the probability of both goals due to the fading function for evidence. Even if this can be an advantage for some applications, this kind of goals which have a common prefix of actions in the plan library reduce the performance of Goldman’s et al. plan recognition system since it has to keep a large number of explanations.
for all the intentions that are consistent with the observed tasks. On the other hand, although both approaches allow us to model the influence of context in the user’s intention, in Goldman’s et al. approach contextual information must be given beforehand and will remain fixed while in our approach they are learned from previous experiences using the user’s feedback. In the same way, our approach facilitates the adaptation of probabilities in the model given the user’s feedback.

The use of Bayesian inference offers many advantages for the task of inferring the user’s intention, but it has also some disadvantages. First, the result of a belief update given some evidence is not sensitive to the order in which the evidence is entered in the network. In other words, if the tasks the agent observes the user performs in the application were given in a completely different order, the resulting confidence levels for non-evidenced nodes would be exactly the same. A second drawback that arises with the use of Bayesian networks is that we are not able to handle repetitive tasks that give rise to cycles.

Finally, our framework demands a considerable effort from the designer of the agent to design the Intention Graph and the complexity of the network is increased since we add extra nodes for representing user preferences. Although the increase in the complexity limits the scalability of our approach, we believe that it is compensated by the gains in terms of early prediction of the user intentions.

References


M. Maragoudakis, A. Thanopoulos, and N. Fakotakis. User modeling and plan recognition under conditions of uncertainty. In Matousek and Mautner, editors, 6th International Conference on Text, Speech and Dialogue (TSD 2003),


Marcelo G. Armentano is an assistant professor in the computer science department and member of the Intelligent Agents Group at ISISTAN Research Institute at UNICEN University in Tandil, Argentina. His research interests include intelligent agents, knowledge management, user modeling, adaptation and personalization. He received a Systems Engineer degree at the UNICEN University in 2003, a Master in Systems Engineer degree at the UNICEN University in 2006 and a PhD degree at the UNICEN University in 2008. Contact him at marmenta@exa.unicen.edu.ar or at ISISTAN, Campus Universitario (B7001BBO) Tandil, Buenos Aires, Argentina.
Analía A. Amandi is a full professor in the computer science department and head of the Intelligent Agents Group at ISISTAN Research Institute at UNICEN University in Tandil, Argentina. Her research interests include intelligent agents and knowledge management. She received a PhD in computer science in the UFRGS, Porto Alegre, Brazil. Contact her at amandi@exa.unicen.edu.ar or at ISISTAN, Campus Universitario (B7001BBO) Tandil, Buenos Aires, Argentina.