SEVENTH FRAMEWORK PROGRAMME

Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	270273
Starting Date:	01-01-2011
Ending Date:	31-12-2015

XPERIENCE.ORG

Deliverable Number:	D4.2.1
Deliverable Title :	Learning Plans for Plan Recognition Using Structural Bootstrapping
Type (Internal, Restricted, Public):	PU
Authors	C. Geib
Contributing Partners	UEDIN

Contractual Date of Delivery to the EC:	31-01-2012
Actual Date of Delivery to the EC:	03-05-2012

Contents

Executive Summary

Plan recognition is a critical component of any system attempting to learn real world plans by observing other agents. In real world domains, previously unseen action sequences to achieve goals of interest will be interleaved with action sequences designed to achieve goals that are already well known. However, for learning, only those action sequences that our current models are unable to adequately explain are of interest. That is, if we already have goal, plan, and action models that explain the actions of the agent, then there is little or no reason to consider learning new goals, plans, or actions from the observed actions. However, in the case where a previously unseen sequence of actions is executed and achieves a goal of interest, then a system, like that being developed in Xperience, may be able to deploy structural bootstrapping to learn these new plans. Thus we can use plan recognition as a filter to remove known plans and allow our learning processes to focus on previously unseen plans for achieving goals of interest.

In the Xperience project, we have proposed to use the ELEXIR plan recognition system[1, 2] to do this filtering. This deliverable deals with extensions to the ELEXIR system that make this possible. Prior to the Xperience project, the ELEXIR system had two significant limitations that would prevent its use in this context:

Propositional representation of action: Prior to the Xperience project, ELEXIR had a strictly propositional representation for actions. For even relatively small domains, this required a large set of actions, one for each possible set of objects the action might work on. This representation is too limited for serious use in an integrated systemlike that in Xperience. As such, it was critical that ELEXIR be extended with a first order representation of actions. ELEXIR has now been extended in this way. For example, where it previously would have required different grasping actions for each of the objects in the world that could be grasped, ELEXIR can now represent all of these actions as a single, generalized grasp action with the object to be grasped as an argument.

Inclusion of state in plan recognition: Prior to the Xperience project, ELEXIR functioned completely within an action space. That is, it searched the space of possible goals to find a plan who's instantiation best matched the observed sequence of actions. The space of possible action sequences is generally much smaller than the full space of possible states of the world and therefore working in this space can make plan recognition search much faster. However, the state of the world can have a significant impact on the plans being executed and even the goals being adopted by the observed agent. Addressing this limitation required adding a first order model of the world, a method of progressing states on the basis of the observed actions, and conditioning ELEXIR's probability computations on the basis of some features of the state model.

The attached paper documents our progress over the last year in addressing both of these limitations in the ELEXIR system. We are happy to report that both of these limitations have been overcome and the ELEXIR system is now expressive enough to be easily used in Xperience domains.

References

[1] Christopher Geib. Delaying commitment in probabilistic plan recognition using combinatory categorial grammars. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1702–1707, 2009.

[2] Christopher Geib and Robert Goldman. Recognizing plans with loops represented in a lexicalized grammar. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11)*, pages 958–963, 2011.

Attached Papers

[Geib:12] Christopher W. Geib., Considering State in Plan Recognition with Lexicalized Grammars., XPERIENCE Technical Report, University of Edinburgh, March 2012.

Abstract: This paper documents extending the ELEXIR (Engine for LEXicalized Intent Recognition) system([1, 2]) with a world model. This is a significant increase in the expressiveness of the plan recognition system and allows a number of additions to the algorithm, most significantly conditioning probabilities for recognized plans on the state of the world during execution. Since, ELEXIR falls in the family of gramatical methods for plan recognition in viewing the problem of plan recognition as parsing, this paper will also briefly discuss how this extension relates to state of the art proposals in the natural language community regarding probabilistic parsing.

Considering State in Plan Recognition with Lexicalized Grammars.

Christopher W. Geib
School of Informatics
University of Edinburgh
10 Crichton Street,
Edinburgh, EH8 9AB, Scotland
cgeib@inf.ed.ac.uk

Abstract

This paper documents extending the ELEXIR (Engine for LEXicalized Intent Recognition) system (Geib 2009; Geib & Goldman 2011) with a world model. This is a significant increase in the expressiveness of the plan recognition system and allows a number of additions to the algorithm, most significantly conditioning probabilities for recognized plans on the state of the world during execution. Since, ELEXIRfalls in the family of *gramatical methods* for plan recognition in viewing the problem of plan recognition as that of *parsing*, this paper will also briefly discuss how this extension relates to state of the art proposals in the natural language community regarding probabilistic parsing.

Introduction

Prior work on the Engine for LEXicalized Intent Recognition (ELEXIR) system (Geib 2009; Geib & Goldman 2011) has a significant limitation. ELEXIR views the problem of plan recognition wholly within the space of possible action sequences. That is, given a sequence of observed actions, ELEXIR is only looking for the plan tree (taken from a library of acceptable plans) whose yield (plan frontier), best fits the observed sequence. In doing this, it only considers the observed actions, not the environment in which they are executed. However, the environment can play a crucial role in differentiating the plans being followed. Consider the following very simple example. Suppose that we observe an agent pullout their cellphone, open it, dial a number and talk to someone. In the absence of information about the state, we would be likely to assume that the person's goal was talking to one of their friends. However, in a world state where we know the agent is standing outside a building that is obviously on fire, we should instead conclude that it is much more likely that the agent was reporting a fire, a very unlikely possibility if the building isn't on fire. Thus, depending on the state of the world when the actions are executed (in this case is the building on fire or not), even the exact same set of actions can result in significantly different probabilities for hypotheses about the plans being executed.

In order to address this problem, ELEXIR must be extended to represent the state of the world and condition its plan hypotheses on the state model. Such conditioning is not a new idea in the literature, although it is a new addition to ELEXIR. Therefore, the rest of this paper will have the following structure. First, it will discuss previous work in context sensitive plan recognition. Next it will provide a brief overview of ELEXIR and its probability model. Next, it will cover extending ELEXIR with a first order predicate state model, and its use within the system's probability model. Finally, because ELEXIR views the problem of plan recognition as one of parsing, it will discuss how this idea is different from the state of the art ideas in probabilistic parsing and the potential for using ideas from this research area in the future.

Prior Work

Graph based, and logical reasoning algorithms have been the basis of a number of previous pieces of plan recognition research(Kautz 1991; Carberry 1990; Litman 1986). Often this work made heavy use of specialized non-probabilistic reasoning about the state of the world . Further, this work made invaluable strides in formalizing the reasoning necessary for plan recognition. However, the general reasoning algorithms and wide ranging inference used in these systems increased the effort required to represent and maintain the domain models and resulted in computationally inefficient algorithms making these systems unusable for real world application.

The last few years have seen a significant increase in the use of probabilistic methods for plan recognition. Much of the current work on probabilistic activity recognition using Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) (Hoogs & Perera 2008; Liao, Fox, & Kautz 2005; Vail & Veloso 2008) is very interesting. The probability computations used in these methods are based on models of probabilistic transitions between states of the world. As such, they generally are able to condition on any information represented in the state description and do not suffer from the same limitation as ELEXIR. However, it is important to recognize that work in this area actually solves a subtly different problem than is being addressed by ELEXIR. Activity recognition focuses on assigning a single, usually unstructured, label to a sequence of observations (for example labeling frames of video). In contrast, ELEXIR solves the problem of combining sequences of such low level activities into higher level plans.

In contrast to work using HMMs and CRFs, the work of Bui and others(Bui, Venkatesh, & West 2002) on Hierarchical Hidden Markov Models does address the same problem as ELEXIR and is able to condition on the state transition models at each layer of the model hierarchy. However, such a close tie to the world state does come at a cost. Their work is unable to consider multiple concurrent and interleaved plans, and has a limited ability to represent plans with loops, both capabilities of ELEXIR.

ELEXIR follows in the footsteps of Vilain's (Vilain 1990) early work on viewing plan recognition as parsing. However this early work does not actually present an algorithm or implemented system for plan recognition. The ELEXIR system is closest in spirit to that of Pynadath and Wellman (Pynadath & Wellman 2000). They provide a method for plan recognition based on probabilistic context free grammars (PCFGs). However unlike ELEXIR, Pynadath and Wellman do not directly parse the observations to produce derivations. Instead, they use the structure of plans captured in a PCFG to build a restricted Dynamic Bayes Net (DBN) to model the underlying generative process of plan construction and execution. They then use the resulting DBN to compute a distribution over the space of possible plan expansions that could be currently under execution. In order to build their DBN, they are only able to handle a limited class of loops, and their treatment of loops must ignore the number of iterations of the loop, two limitations not shared by ELEXIR.

Instead of these methods, the work described here will follow the current work in probabilistic natural language parsing by conditioning the application of gramatical rules used to define the set of acceptable plans (and hence the possible hypothesis) on the state of the world. In order to see this, the next section will briefly review the ELEXIR system.

Plans as Grammars in ELEXIR

Following other work on *gramatical methods* for plan recognition, ELEXIR(Geib 2009) views the problem as one of *parsing* a sequence of observations, based on a formal grammar that captures the possible plans that could be observed. In the case of ELEXIR, the plans are represented using a particular *lexicalized grammar* called Combinatory Categorial Grammars (CCG) (Steedman 2000). Unlike more traditional grammars, like context free or regular grammars, where the constraints specific to a language are spread between the rules of the grammar and the lexicon, lexicalized grammars move all language-specific information into rich data-structures called *categories*. Such categories are associated, by the lexicon, with individual observables (the *terminals* of traditional grammars). Further lexicalized grammars and use a small set of language independent rules to combine lexical categories for parsing. Thus, parsing lexicalized grammars abandons the application of multiple grammar rules in favor of assigning a lexical category to each observation and combining the categories to build a parse.

To represent a set of possible plans in a CCG, each observable action is associated with a set of syntactic *categories*, defined recursively as:

Atomic categories : A finite set of basic action categories. $C = \{A, B, ...\}$.

Complex categories : $\forall Z \in C$, and non empty set $\{W, X, ...\} \subset C$ then $Z\backslash\{W, X, ...\} \in C$ and $Z/\{W, X, ...\} \in C$.

Complex categories represent functions that take a set of *arguments* ($\{W, X, ...\}$) and produce a *result* (Z). The direction of the slash indicates where the function looks for its arguments. We require the argument(s) to a complex category be observed after the category for forward slash, or before it for backslash in order to produce the result.

To provide some intuitions, the semantics of the \ is roughly that of a sub-plan that must precedes the present category and the semantics of / is sub-plan that must succeed the current category to produce the result. Therefore, an action with category $A\backslash\{B\}$ is a function that results in performing a plan for A when a plan with category B has already been performed. Likewise, $A/\{B\}$ is a function that results in performing A if a plan with category B is executed later.

Consider the phone call example. A very simplified lexicon for the observable actions of getting the phone, opening it, placing a call, and talking could be:

CCG: 1

$$\begin{aligned}
dial(Cellphone) :=& ((REPORT/\{T\})\backslash\{G\})\backslash\{O\} \mid \\
& ((CHAT/\{T\})\backslash\{G\})\backslash\{O\}. \\
talk(Cellphone) :=& T. \\
get(CellPhone) :=& G. \\
open(CellPhone) :=& O.
\end{aligned}$$

Where $G, O, T, REPORT$, and $CHAT$ are basic categories, and the observation of the action *dial* has two complex categories assigned to it by the lexicon: one for reporting a fire and one for chatting to a friend. Note, that each action in this simplified example lexicon has a single variable argument. Therefore any observed action instance will have bound such variables (ie. the variable argument "Cellphone" will be bound to some particular object).[1]

ELEXIR uses three *combinators* (Curry 1977) defined over pairs of categories, to combine CCG categories into higher level plan structures:

$$\begin{array}{lll}
\textit{rightward application:} & X/\alpha \cup \{Y\},\ Y & \Rightarrow\ X/\alpha \\
\textit{leftward application:} & Y,\ X\backslash\alpha \cup \{Y\} & \Rightarrow\ X\backslash\alpha \\
\textit{rightward composition:} & X/\alpha \cup \{Y\},\ Y/\beta & \Rightarrow\ X/\alpha \cup \beta
\end{array}$$

where X and Y are categories, and α and β are possibly empty sets of categories. To see how a lexicon and combinators parse observations into high level plans, consider the derivation in Figure 1 that parses the observation sequence: $get(obj1), open(obj1), dial(obj1), talk(obj1)$ using CCG:1. Note that each observation's argument is now bound to a unique object identifier. As each observation is encountered, it is assigned a category as defined in the lexicon. Combinators (rightward and leftward application in this case) then combine the categories.

To enable incremental parsing of multiple interleaved plans, ELEXIR does not use an existing parsing algorithm.

[1]The action arguments are largely a distraction to this portion of the discussion and they could be left off for clarity here. However, since they will be used later in our discussion they are shown.

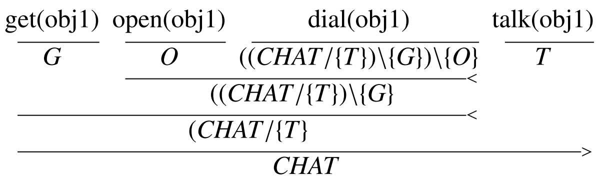
get(obj1)
open(obj1)
dial(obj1)
talk(obj1)
G
O
$((CHAT/\{T\})\backslash\{G\})\backslash\{O\}$
T
<
$((CHAT/\{T\})\backslash\{G\}$
<
$(CHAT/\{T\}$
>
$CHAT$

Figure 1: Parsing Observations with CCG categories

Instead it uses a very simple two step algorithm based on combinator application linked to the in-order processing of each observation.

- Step 1: Before an observation can be added to the explanation with a given category, it must be possible to use leftward application to remove all of the category's leftward looking arguments. We call such categories *applicable* given the observed action and current explanation.
- Step 2: After each of the applicable categories for the observation has been added to a copy of the explanation, ELEXIR applies all possible single combinators to each pairing of the new category with an existing category.

This two step algorithm both restricts observations to take on only applicable categories, and guarantees that all possible combinators are applied. At the same time, it does not force unnecessarily eager composition of observations that should be held back for later use.

For the rest of this discussion it will be helpful to have defined a few terms. First a *parse* or *explanation* of a sequence of observations is just an ordered sequence of (possibly complex) categories that result from executing the two step algorithm just described. Each parse of the observations represents a distinct set of categories, or methods for combining the categories, that can be used to account for the observed actions. As such they "explain" the actions. For the reminder of this paper we will use the terms "parse" and "explanation" interchangeably.

For each of the categories in an explanation we can define the *goal* or a *root result* of the category. In the case of basic categories it is the category itself. For complex categories it is the left-most inner result category. For example, we will refer to *REPORT* as the goal or root result of the category:

$$((REPORT/\{T\})\backslash\{G\})\backslash\{O\}$$

With these tools in hand, we can imagine using a CCG to define the set of acceptable plans, and the two step algorithm to generate a set of possible explanations for a given set of observed actions. However, in order to determine the probability of a given goal these explanations must have an accompanying probability model.

The Probability Model

Suppose that ELEXIR has built a complete an covering set of explanations for the observations resulting in a number of explanations. ELEXIR computes the conditional probability of each goal as:

Definition 1.1

$$P(goal|obs) = \frac{\sum_{\{exp_i|goal \in exp_i\}} P(exp_i \wedge obs)}{\sum_{j=0}^{n} P(exp_j \wedge obs)}$$

where $P(exp_i \wedge obs)$ is the probability of explanation exp_i and the observations. Thus, the conditional probability for any particular goal is the sum of the probability mass associated with those explanations that contain it divided by the probability mass for all the explanations that can result from the observations. This critically relies on computing the probability for each explanation.

For an explanation, exp, of a sequence of n observations, $\sigma_1...\sigma_n$, that results in m categories in the explanation, ELEXIR computes the probability of the explanation as:

Definition 1.2

$$P(exp \wedge \{\sigma_1...\sigma_n\}) = \prod_{i=1}^{m} P(root(c_i)) \prod_{j=1}^{n} P(cinit_j|\sigma_j)K$$

Where $cinit_j$ represents the initial category assigned in this explanation to observation σ_j and $root(c_i)$ represents the root result category of the ith category in the explanation(exp) and K is a normalizing constant.

The second term captures the likelihood of each observation being given the lexical category it has been initially assigned given the set of possible categories the lexicon allows it to have. This is standard in natural language parsing and assumes the presence of a probability distribution over the possible categories each observation can have in the lexicon. In prior work, this term has been modeled by a uniform distribution across all the possible categories (all the alternatives are considered equally likely.)

The first term is the prior probability of the agent having the root goals present in the explanation. This is the probability that the current categories in the explanation are the agent's root goals, and they will not be combined into a larger goal. Note that this term is not included in traditional natural language parsing. For natural languages all input streams are assumed to be a single sentence. Therefore all observation streams result in a single common nonterminal (usually "S" for sentence), and therefore, this kind of probability term is not included in the parsing model. However, since ELEXIR supports recognizing multiple, interleaved plans such a term is a necessary part of the probability model.

Considering State

Taking context into account requires extending both of the terms of Definition 1.2 to condition on the state of the world. The question is which state of the world? There are compelling reasons to consider, the state before any of the observations have been performed, one of the intermediate states during the execution of the observed actions, and the state after the last observed actions. In this work we will actually argue for different answers for the two different terms in Definition 1.2. We consider each of them in turn.

$P(root(c_i)$ As we have already argued, one of the stronger pieces of evidence for the goals of the agent is the state of the world before the plan is executed. More often than not, the agent will be attempting to deliberately change the state of the world. Thus, conditioning on the final state, or any of the intermediate states would be inappropriate. If the agent has been successful in its plans the state might already have changed to that desired by the agent. Therefore, for the probability term for the root results of an explanation, we will condition on the initial state of the world resulting in:

$$\prod_{i=1}^{m} P(root(c_i|s_0))$$

Where s_0 denotes the state of the world before any observed actions. Note that, ELEXIR's ability to recognize multiple interleaved goals weakens this argument slightly. If an agent has multiple goals, it is not at all clear that they were all adopted at the same time. For example, consider an agent with a goal, G_0, that it has started executing a plan, P_0, to achieve. It is entirely possible that a state that occurs during the course of the execution of the plan P_0, call it $s_{i>0}$, leads the agent to adopt a new goal, G_1. Consider the case of where an agent is planning on chatting to a friend. They pick up the phone and open it. While preparing to call their friend, they notice the building in front of them is on fire and then adopt the goal of calling the fire department to report the fire. In this case, conditioning on the initial state of the world would suggest that the agent was calling a friend because the existence of the fire is not represented in the initial state.

We can also imaging cases where the agent adopts a goal and even a particular plan to achieve it, but does not begin execution until much later. This would argue for conditioning on a state of the world that occurred significantly before the beginning of the observed action sequence. In general, this would argue for considering the process of goal adoption in the agent, and conditioning the probability of the root goal on the state that caused its adoption. Consider agreeing to go to a concert several weeks in advance with a friend. The goal is developed quite early, but it is not acted on until much later. However, since a full treatment of goal adoption is outside the scope of this paper, we will simplify these considerations and assume that the agent acquires all of the goals they are pursuing immediately before observations begin. Therefore, we will condition all goals on the initial state before any actions are observed, and simply note as future work that this could be changed given a model of goal adoption.

$P(cinit_j|\sigma_j)$ This term is already conditioned on the observed action, but we can relatively easily imagine conditioning on the state of the world as well. Consider the case where in we observe an agent picking up an umbrella, and we know there are two plans that involve picking up the umbrella: 1) preparing to go out in the rain 2) cleaning up by putting the umbrella away, say in the closet. Further, suppose as a result of the two plans, the plan lexicon has two different categories for picking up the umbrella, one for each of the two plans. In the absence of any other information, we might assume that each of the possible categories are equally likely. However, if we notice that the umbrella is already in the closest when the picking up action occurs, then it should be very unlikely that the agent is cleaning and the category for going out in the rain should be considered more likely. Likewise if it were raining out and the agent is wearing a coat again our intuition suggests that both plans are not equally likely. It is more likely that the agent is going outside, and we would want to condition the probability of the category we chose for the observed action on the state of the world.

This argues for conditioning on the state immediately before execution of the action as well as the observed action itself. This results in:

$$\prod_{j=1}^{n} P(cinit_j|\sigma_j, s_j)$$

where s_j denotes the state of the world immediately before the execution of the ith action. Bringing this together with the previous term results in rewriting Definition 1.2 as:

Definition 1.3

$$P(exp \wedge \{\sigma_1 ... \sigma_n\}) = \prod_{i=1}^{m} P(root(c_i|s_0)) \prod_{j=1}^{n} P(cinit_j|\sigma_j, s_j)K$$

The number of examples given here suggests the possibility of conditioning these probability computations (and thereby the associated choices in the explanation) on a variety of different problem features. We will return to discuss this further, but first we need to discuss the practical impact of using state within these computations.

Implementational Considerations

The kind of conditioning described above requires access to the initial state of the world in order to condition the root goal,s and it requires access to the state of the world before each of the observed actions to condition the category selection. A straightforward method to achieve this would be to extend the defined inputs to the plan recognition problem to include complete observations of the state of the world before each of the observed actions (the initial state simply being the state before the first observed action).

However, instead of requiring complete observations of the state before each action, we have extended ELEXIR to maintain its own internal model of the world. To do this we require as input to the plan recognition problem, a model of the initial state of the world, and a set of precondition-effect rules for each action. When processing each action, the precondition-effect rules and the previous state can be used to produce a model of the following state (the state after the action is executed). This allows ELEXIR to maintain its own model of the state of the world and condition its explanations based on this.

This was done so that plan recognition could be performed in situations where constant monitoring of the state of the world is impractical either for engineering or physical reasons. For example, consider plan recognition in a computer network security domain. While collecting audit logs for a system is a common occurrence, storing a complete trace of the state of the entire network over the entire window of the audit would be prohibitively large. On the other hand, not all of the state information will be needed by ELEXIR to conditionalize. Frequently only a small number of predicates will be needed to condition the explanation's probabilities correctly. In light of this, we believe that by careful modeling of the initial state of the world and the observable actions, we can both limit the size and cost of the state model being maintained for conditionalization while still providing a model that is rich and accurate enough to improve the results of plan recognition. With this in mind, extending ELEXIR to make use of state requires extending the knowledge contained in the domain description as well as the addition of more computational machinery. We discuss each of these in turn, with examples to provide intuitions about their use.

Required Additional Domain Knowledge

Extending ELEXIR to account for context requires the additional of the following four pieces of domain knowledge.

1) Model of the initial state of the world: As we have pointed out, in order for ELEXIR to maintain a model of the state of the world as it evolves under the effects of the observed actions, the system must be provided with a model of the initial state from which to work. The representation used for state models in this work is a limited first order logical representation. The space of all possible state models is defined by a set of terms built up from a closed set of predicates (representing the relations and properties that can be observed in the world) and a possibly empty list of arguments representing the objects in the world (again taken from a closed list). A model of a state of the world is defined by a list of such terms that enumerate those specific relations and properties that are true in the world. Note the terms in these models are implicitly related by conjunction, and this work will assume a closed world. Therefore and any term not explicitly stated to be true in the model will assumed to be false.

We acknowledge, that this is a particularly limited and well known representation for state models. For example it can not support states with disjunction or even explicit negation of a term. There are much more expressive representations that could be used, however, logical expressiveness often comes at a high computational cost both for computing model state progression and determining if a given term is true in the state. Since the purpose of this work is to demonstrate the viability of using state in such systems, the use of more expressive state modeling languages is left as an area for future work.

Consider the cell-phone calling example. Part of a very simplified initial state model could be:

$$[fire, handEmpty, cellphone(obj1), off(obj1), ...]$$

Which among other things would tell us that there was a fire, the agent's hands were empty, that object1 is a cellphone and that it is currently off.

2) Precondition-effect rules for each of the actions: For each observable action, ELEXIR will require a set of precondition-effect rules that defines how the modeled world state evolves. In these precondition and effect rules we extend the representation of states with negation for individual terms. That is, in any given rule, a precondition or effect is defined as a set of, possibly negated, terms. In the case of preconditions, non-negated terms must be listed in the modeled world state and negated terms must not occur in the modeled world state for the rule to be applicable.

Like traditional STRIPS rules(Fikes & Nilsson 1971) for action projection, the terms in such a precondition can also have variables as arguments that are bound either by coreference to one of the observations arguments or when the precondition is ground in the state model. Such variables can be used to guarantee that the same argument plays a role in multiple terms in the precondition. Also like traditional STRIPS rules, if the condition is satisfied in the state model, then to create a model of the state that results from executing the action, all of the negated terms in the effect are removed from the current state model and all of the non-negated terms are added. This results in a new state model that captures what should be true after the execution of the action.

For example, consider the following rule for the action *openP* with an argument X where the precondition is the first set of terms and the effect is the second :

$$open(X) : [cellphone(X), off(X)], [!off(X), on(X)]$$

It specifies that if *openP* is performed on an object that is a cellphone and that cellphone is initially off that in the resulting state of the world, off is no longer true of the cellphone and *on* is now true of the cellphone. Note, that we assume the precondition-effect rules for each action have non-overlapping preconditions such that only one rule's precondition is true in any given state model.

3) Probability distributions for the root categories of the explanation conditioned on the state model: For each possible root category, ELEXIR must have access to $P(root(c_j)|s_0)$. Therefore, domain specification must now provide a set of *root probability rules* to define the conditional probability distribution for each possible root result. Each such rule is defined by a pair containing a condition and a probability. The condition defines those state models for which the rule holds, and the probability defines the conditional probability of the agent having the category as one of its goals in worlds where the condition is true. Like the precondition-effect rules, the conditions in these rules are allowed to use negated as well as unnegated terms.

The following example displays two root probability rules for each of $REPORT$ and $CHAT$:

$$REPORT :([fire], 0.99), ([!fire], 0.01).$$
$$CHAT :([fire], 0.01), ([!fire], 0.99).$$

These rules capture the idea that reporting a fire is much more likely as a goal of the agent in world states where there is a fire than in states were there is no fire. Likewise, calling a friend is much less likely in a state with a fire than a state without a file. Note we also allow the domain designer to specify a default prior probability if none of the conditions apply.

4) Distribution of the possible categories an observation can take on: For each possible observable action ELEXIR needs to know $P(cinit_i|\sigma_i, s_i)$. The domain designer does this through a set *category assignment rules* made up out of a condition and a probability distribution over the possible categories. In each such rule, a condition is specified by a set of negated and unnegated terms and the distribution specifies the likelihood of each category given that the condition is satisfied in the modeled state.

For example, consider the following category assignment rules:

$$
\begin{aligned}
dial(X) : (&([fire], [((REPORT/\{T\})\backslash\{G\})\backslash\{O\} = 0.9,\\
&\qquad ((CHAT/\{T\})\backslash\{G\})\backslash\{O\} = 0.1])\\
&([!fire], [((REPORT/\{T\})\backslash\{G\})\backslash\{O\} = 0.1,\\
&\qquad ((CHAT/\{T\})\backslash\{G\})\backslash\{O\} = 0.9]))
\end{aligned}
$$

These two rules specify that in the case of fire, the action of dial is much more likely to have *REPORT* as its root result than *CHAT*, and the probabilities are reversed if there is no fire. Note that the probabilities specified in each rule sum to one.

Again we assume that the set of conditions for these rules are non-overlapping so that at most one of them is satisfied in any model state. If no condition-distribution rules are provided, or none of the conditions matches the current world state, then the system defaults to a uniform distribution over the actions categories.

Required Additions to ELEXIR's Algorithm

The use of these four pieces of domain knowledge is relatively straightforward. Three steps are added to the existing two step parsing algorithm. First we initialize the state model with the initial state provided for the problem, and then for each observation we executing the following algorithm:

- begin loop

  Step 1: Clone the explanations and apply all applicable categories that are consistent with the next observed action.

  Step 2: Apply all possible single combinators to each pairing of the new category with an existing category.

  Step 3 (new): To compute the probability of the observation being given the category, search the category assignment rules for a rule satisfied in the current state model. That rule then specifies the probability distribution for each of the categories used for this action.

  Step 4 (new): The precondition-effect rule for this action that is satisfied in the current state model is used to produce a new state model.
- end loop
- Step 5 (new): Once all of the observed actions have been processed, we return to the initial state model and use the root probability rules to determine the conditional probability of each of the root results of the explanation.

While the possible benefits of using this algorithm are significant for some domains, there is a computational cost of maintaining a world model of the kind described here, and conditioning the probability of explanations and goals on it. We note that while the system as described here has been implemented, and anecdotal evidence suggests that the additional cost of maintaining such a state model and conditioning on it is not significant, we do not yet have enough empirical evidence to back up this claim for real world sized problem. We are currently in the process of testing of these extensions to the ELEXIR system to determine the costs associated with this additional machinery.

Relation to Probabilistic Parsing

Recent prior work in probabilistic parsing for natural language, has considered larger numbers of conditioning variables in the grammatical model(Collins 1997) and even using such probabilistic models with CCGs(Hockenmaier & Steedman 2002). This work suggests that in the action context there are at least four possible things such grammars could be conditioned on:

1. The observed action: this was part of the initial formulation of the of the ELEXIR system.
2. The state of the world: this is the addition to ELEXIR presented here.
3. The previously observed actions: this would allow the system to condition the choice of a syntactic category for one action based on, possibly multiple, previously observed actions.
4. The categories assigned to previous actions: Like the previous case, this would allow ELEXIR to condition its choice of the current category based on the categories chosen for previous actions. It is worth noting that that this would effectively allow the system to arbitrarily condition on the existing structure of the currently hypothesized plans. For example: we can imagine that the probability that an agent will take more objects on a short trip could be conditioned on a prior decision to drive their own car, rather than walking or taking the bus.

To the best of our knowledge, there are no parsing based plan recognition systems that have used the last two of these possibilities, but there is every reason to suppose they will be as effective in plan recognition as they are at natural language parsing. Thus, while the current work does push forward the work on the ELEXIR system, this related work already highlights future directions for work in plan recognition.

Conclusions

This work extends the ELEXIR probabilistic plan recognizer with a model that takes the state of the world into consideration. ELEXIR is based on parsing the observations in a maner very similar to that of natural language processing. We have extended its probabilistic algorithm to conditioning the probability of the possible parses on the state. It has also briefly described how the state propagation is handled, and finally it has outlined future directions in the form of other possible conditioning variables in the natural language literature that could be considered in this domain.

References

Bui, H. H.; Venkatesh, S.; and West, G. 2002. Policy recognition in the Abstract Hidden Markov Model. *Journal of Artificial Intelligence Research* 17:451–499.

Carberry, S. 1990. *Plan Recognition in Natural Language Dialogue*. ACL-MIT Press Series in Natural Language Processing. Cambridge, MA: MIT Press.

Collins, M. 1997. Three generative, lexicalised models for statistical parsing. In *ACL '97: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*.

Curry, H. 1977. *Foundations of Mathematical Logic*. Dover Publications Inc.

Fikes, R., and Nilsson, N. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Geib, C., and Goldman, R. 2011. Recognizing plans with loops represented in a lexicalized grammar. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11)*, 958–963.

Geib, C. 2009. Delaying commitment in probabilistic plan recognition using combinatory categorial grammars. In *Proceedings IJCAI*, 1702–1707.

Hockenmaier, J., and Steedman, M. 2002. Generative models for statistical parsing with combinatorial categorial grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 335 – 342.

Hoogs, A., and Perera, A. A. 2008. Video activity recognition in the real world. In *Proceedings AAAI*, 1551–1554.

Kautz, H. A. 1991. A formal theory of plan recognition and its implementation. In Allen, J. F.; Kautz, H. A.; Pelavin, R. N.; and Tenenberg, J. D., eds., *Reasoning About Plans*. Morgan Kaufmann. chapter 2.

Liao, L.; Fox, D.; and Kautz, H. A. 2005. Location-based activity recognition using relational Markov networks. In *Proceedings of IJCAI*, 773–778.

Litman, D. 1986. Understanding plan ellipsis. 619–626.

Pynadath, D., and Wellman, M. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings UAI*, 507–514.

Steedman, M. 2000. *The Syntactic Process*. MIT Press.

Vail, D. L., and Veloso, M. M. 2008. Feature selection for activity recognition in multi-robot domains. In *Proceedings AAAI*, 1415–1420.

Vilain, M. B. 1990. Getting serious about parsing plans: A grammatical analysis of plan recognition. In *Proceedings AAAI*, 190–197.