A Generic Formal Plan Recognition Theory

Guy Camilleri
IRIT – SMAC Université Paul Sabatier,
118 Route de Narbonne,
31062 Toulouse Cedex France
camiller@irit.fr

Abstract

Plan recognition is the task of inferring intentions (in terms of plans) from the actions or utterances of observed actors (Kautz [11]). In this paper, we provide a new logical formalization of plan recognition, which allows us to clearly specify, validate and design the plan recognition process (useful in several domains such as: discourse understanding, Human Computer Interaction, and so on).

However, depending on the applications different recognition strategies are used (top-down, bottom-up, and so on). Therefore, to describe the possible strategies, we only chose to formalize the reasoning basic steps of plan recognition. Moreover, only the specific aspects of the recognition problem are formalized and fixed. Therefore, the other aspects can be determined to satisfy the system needs.

To illustrate our formalism, we describe two strategies of plan recognition used in many applications. We also present an algorithm useful to the discourse understanding.

1. Introduction

Plan recognition defined by Kautz in [9,10,11] is the task of inferring intentions (in terms of plans) from the actions or utterances of observed actors. This consists in explaining the observations by finding the used plan that contains (or entails) these observations. The plan recognition context studied in this paper is the keyhole plan recognition context where the actors are unaware of or indifferent to, the plan recognition process (as if you are looking through a keyhole).

The plan recognition problem is studied in Artificial Intelligence in a variety of research domains. Some of these domains include spoken dialogue [2], Human Computer Interaction [13,18,20], discourse management [14,15,16], collaborative (cooperative) systems [13,18], intelligent tutoring and help [20]. The plan recognition process used in these domains generally differs. However, a keyhole plan recognizer is always used to establish the link between the domain plan (under construction) and the observed actions. Usually, the keyhole recognizer is a part of a more global plan recognizer [13,15]. For example, in the discourse understanding context, the discourse structure is maintained by a global plan recognizer using a keyhole plan recognition process to determine the bind between the action conveyed by an utterance (observed action) and the domain plan.

In the literature, several works propose a framework for keyhole plan recognition [4,9,10,11,12]. The utilization of a logical formalization provides a very clear specification, which permits us to extend and modify the algorithm. The best work which makes clear and explicit the dissociation between the plan recognition reasoning and the other aspects involved, is the Kautz's formal theory [9,10]. Unfortunately, the Kautz's theory of keyhole plan recognition only deals with bottom-up recognition, which starts from the observed action. However, most theories of discourse understanding use a top-down recognition process to take into account the discourse context [7,13,15,16].

The main purpose of this contribution is to define a powerful logical formal theory of keyhole plan recognition which can be used to specify every kind of plan recognition process (top-down, bottom-up, mixed, …). We describe in logic the basic inference steps of our plan recognition theory. The way to use these steps provides a clear specification of algorithms. Therefore, we present a theory which can be used to specify, validate and test a set of algorithms. The theory inherits from Kautz's theory an essential generic nature.

First, we present the context and the application field of our plan recognition theory. We then expose our formalization by defining the concepts handled and the assumptions modeling the plan recognition behaviors. We then illustrate two plan recognition strategies by examples, and then present an algorithm of plan recognition useful to discourse understanding. Finally, we discuss some relevant related works and the future work.

2. Overview

The required assumptions to apply our keyhole plan recognition theory are the closed-world (or circumscripted world) assumption and the correctness assumption.

The closed-world assumption states that the known ways of performing a task (or action) are the only ways of performing that task (or action). Therefore, for all observations, a known way, which explains observations exists. This assumption can appear too strong, but it seems reasonable to consider that an agent can only reason on the knowledge that he currently possesses. Likewise, he can acquire further knowledge but at a given reasoning point he can only use the knowledge acquired.

The correctness assumption claims that the plan recognition only finds the correct plans which explain observations. The correctness property is defined via the plan library and the satisfaction of constraints. Indeed, the plan found by the plan recognizer is the result of a correct (or consistent) utilization of the plan library. Furthermore, if the plan library contains some incorrect plans or if it uses a module which allows inconsistent constraint satisfaction then the recognition process is able to recognize some erroneous plans [17].

With these assumptions, the proposed plan recognition theory can manage multiple plans. Often, several plans (or multiple plans) can explain an observation. This implies that the plan recognition system must maintain several plans until a new observation eliminates some candidate plans. For example, if an observation can be explain by two plans P1 and P2, the system must consider at this point of discourse only these two possibilities. Suppose then another observation arises and this observation can only be a part of (or explained by) P1. Therefore, the plan recognizer can reasonably deduce that the sole plan, which can explain these two observations, is P1. This kind of deduction is possible only if the closed-world assumption is used. Furthermore, the correctness assumption establishes that the found plans are consistent about the plan library and the constraint satisfaction process.

Moreover, most theories of discourse understanding use the notion of focalized task. This focalized task allows us to specify the recognition context from which we intend to explain observations.

Our theory does not make any presumption about the temporal relationship between tasks (or actions) or about the consistency checking mechanism between states (or a task's effects and preconditions). This allows us to manage non-sequential tasks (or actions), temporal constraints and consistency tests dependent of application needs.

This generic feature allows us to use the adapted temporal constraints and state manager to satisfy the application needs. Another advantage of constraint independence is that it increases the system flexibility because it permits us to change dynamically the kind of the temporal constraints and state manager in the same application.

3. Formalization

We chose to formalize our plan recognition theory in second-order logic with equality, but a great part of our formalization is only described in first-order logic. However, only the basic reasoning steps are formalized, we do not make any assumptions on the way to use them. This increases the representative power of our formalism. Indeed, the different algorithms of plan recognition are performed by a particular inference engine, which determines the application order (and context) of the basic reasoning steps. In fact, an algorithm is viewed as an ordered set of basic reasoning steps. This kind of formalization is very useful to design, validate and test algorithms.

3.1. Concepts used by plan recognition

3.1.1. Task. A task is an action (or more exactly an action denomination) in particular context. A task context is all world objects related to this task. Usually the context is represented by parameter list, which can be empty. For example, the "Joe eats apple" task is modeled by Eat(Joe, Apple) and the list (Joe, Apple) describes the context of the task.

We borrow our task representation from Kautz's event formalization. The tasks are domain elements. We note the interpretation domain D. Therefore, in our logical language L, the tasks are represented by terms (constant, variable or function symbols). The task Types are represented by unary predicates and serve to identify the domain elements which are tasks. So, I(Type(x)) is true when I is an interpretation and x a task of type Type. In the above example, if I(EAT(x)) is true then x is a task of EAT type.

The task parameters are accessible through the role functions and are described by function terms in L (which apply on a task and return a parameter), and are interpreted by a domain function (defined in $D \rightarrow D$). For example, if x is a task then the role function agent(x) denote the domain element which is the agent of x. Various role functions are used on a task to access the different parameters, including its time.

3.1.2. Recipe. The recipe notion introduced by Grosz [5] describes a task achievement representing the hierarchical decomposition (at one "abstraction" level) of a task achievement under appropriate constraints. A recipe describes a way of task execution. A recipe for the task T is thus a triplet made up of T, a subtask set $\{T_{11}; \ldots;T_{1n}\}$ and a constraint set $\{C_1; \ldots;C_m\}$, noted $(T,\{T_{11}; \ldots;T_{1n}\},\{C_1; \ldots;C_m\})$.

Consider the following recipe in the cooking micro-world (adapted from Kautz [9,10]):

Let an agent A1 which achieves in the kitchen:
"boils some water" at time T1,
"adds the pasta" at T2,
"waits ten minutes" at T3,
"makes the sauce" at T4,
"mix the pasta with the sauce" at T5,
"makes a pasta dish" at T6.

The constraint set is divided in four subsets (see Kautz [10]).
Precondition subset describing the world State required to apply the recipe (or the initial state).
Effect subset representing the world state generated by the recipe execution (or the final state).
The *equality* subset modeling the equal relationship between the parameters of the recipe tasks.
The *temporal* subset representing the temporal relationship between tasks.

In the previous example, the precondition subset could be "In the kitchen, there are at least two pans, some pasta, and so on". The effect subset could be "the pan is used, the pasta dish is made, and so on". The equality subset could claim that the same agent must perform all tasks in the recipe, in the same kitchen. The temporal subset could express the following execution sequence: T1 before T2 and so on.

A complete representation of the recipe definition proposed above requires a temporal logic. Indeed, in this definition, the "achievement" task property is present. To represent the performed task, we can use the Occur operator (defined by Allen [1]), the DO operator used by Grosz [6], or the DONE operator introduced by Cohen and Levesque [3]. We choose for the moment Grosz's DO operator to model the recipe notion.

A recipe for a type task A is composed of subtasks type $\{B1; \ldots;Bn\}$ $n \in \mathbf{N}$, under the constraints Γ is represented by a logical entailment (or recipe axiom) as :

$\forall x\ DO(B_1(s_1(x))) \wedge DO(B_2(s_2(x))) \wedge \ldots \wedge DO(B_n(s_n(x\ ))) \wedge \Gamma \supset DO(A(x))$

Where A,B1, …,Bn represent predicates of task type
and,$s_1(x),\ldots,s_n(x)$ describe some decomposition role functions and,
Γ represent the conjunctive set of constraints (therefore, the formula resulting is the conjunction (and $\wedge$) of the constraint's formulas).

The decomposition role functions are used to access the subtasks. Therefore, the decomposition role functions return a task. Thence, the above entailment states that if $s_1(x)$ task of type B1 is performed and the $s_2(x)$ task of type B2 is done and $s_n(x)$ task of type Bn is achieved and the constraints Γ hold, then the task x of type A is performed. Note that the number of the decomposition role functions do not express a particular order.

For our purpose, the temporal operator DO can be omitted because when we treat or formulate a task type predicate, we always consider its achievement. The recipe presented above can be then formalized in the following way:

$\forall$ x

Subtasks BoilWater($s_1(x)$) $\wedge$ AddPasta($s_2(x)$) $\wedge$
Wait($s_3(x)$) $\wedge$ MakeSauce($s_4(x)$) $\wedge$ Mix($s_5(x)$)$\wedge$

Equality constraints agent(x)=agent($s_1(x)$) $\wedge$ agent(x)= agent($s_2(x)$) $\wedge$ agent(x)=agent($s_3(x)$) $\wedge$ agent(x)=agent($s_4(x)$) $\wedge$ agent(x)= agent($s_5(x)$) $\wedge$ kitchen(x)=kitchen($s_1(x)$) $\wedge$ kitchen(x)= kitchen($s_2(x)$) $\wedge$ kitchen(x)= kitchen($s_3(x)$) $\wedge$ kitchen(x)= kitchen($s_4(x)$) $\wedge$ kitchen(x)= kitchen($s_5(x)$) $\wedge$

Temporal Constraints Started-by(time(x),time($s_1(x)$)) $\wedge$Before(time($s_1(x)$),time($s_2(x)$)) $\wedge$Before(time($s_2(x)$),time($s_3(x)$)) $\wedge$Before(time($s_3(x)$),time($s_4(x)$)) $\wedge$Before(time($s_4(x)$),time($s_5(x)$)) $\wedge$Finished-by(time(x),time($s_5(x)$)) $\wedge$

Preconditions Available(pan1(kitchen(x))) $\wedge$Available(pasta(kitchen(x))) $\wedge$Available(pan2(kitchen(x))) $\wedge$ Available(vegetables(kitchen(x))) $\wedge$

Effects Used(pan1(kitchen(x))) $\wedge$ Used(pan2(kitchen(x)))$\wedge$Used(pasta(kitchen(x))) $\wedge$ Used(vegetables(kitchen(x)))

$\supset$ MakePastaDish(x)

3.1.2. Task Type Hierarchy.

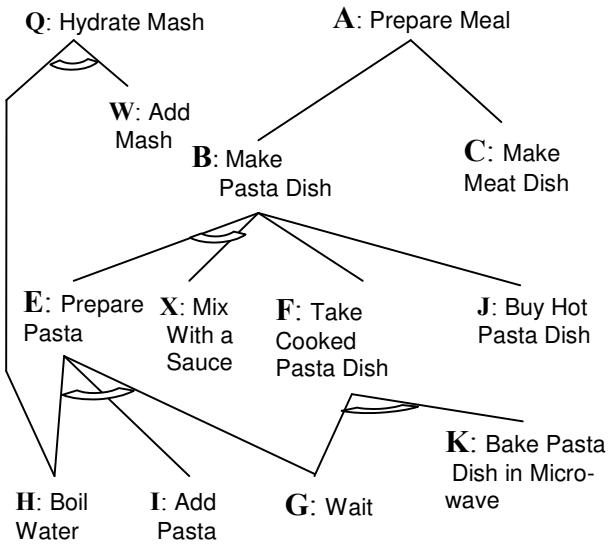

Figure 1: Partial cooking world hierarchy

The task type hierarchy is a collection of recipes hierarchically organized, representing all possible decompositions (following the described recipes) of task performance. It is thus represented by a conjunctive set of recipe axioms.

Consider the following task type hierarchy in the cooking micro-world:

In figure 1, the non-oriented arcs represent the decomposition relationship between task types. A circle arc groups the task types belonging to the same recipe. For example, the task type "Make Pasta Dish" referenced by the letter B owns three recipes (Make Pasta Dish, {Prepare Pasta; Mix With A Sauce},{…}), (Make Pasta Dish,{Take Cooked Pasta Dish from Freezer},{..}) and (Make Pasta Dish,{Buy Hot Pasta Dish},{…}). We do not represent the constraint sets in figure 1.

This task type hierarchy is formalized by the conjunctive H set which contains all recipe axioms. Thence, the H set describing the above hierarchy is:

$H= \{\forall x\ B(s_1(x)) \wedge \Gamma_1 \supset A(x);\ \forall x\ C(s_2(x)) \wedge \Gamma_2 \supset A(x);$
$\forall x\ X(s_2(x)) \wedge E(s_1(x)) \wedge \Gamma_3 \supset B(x);\ \forall x\ F(s_3(x)) \wedge \Gamma_4 \supset B(x);\ \forall x\ J(s_4(x)) \wedge \Gamma_5 \supset B(x); \forall x\ \Gamma_{14} \supset W(x);\ \ldots\}$

Where Γ_i with $i \in [1,14]$ describes the recipe constraint sets.
We also define the following sets:
H_T ={task type predicates}, in the above example H_T={A; B; C; E; X; F; J; H; I; G; Q; W; K}
H_R={ root task type predicates} = {Q; A}
H_E={ elementary (or basic) task type predicates} = {H; I; G; K; J; W}

The leaf task types in the previous diagram represent the elementary task type. No subtasks are described for this kind of task types. Nevertheless, these task types own a recipe (for their achievement) with an empty subtask set.

3.2. Plan recognition assumptions

Our keyhole plan recognition theory is described with the following assumption set. These assumptions model the basic reasoning steps of the recognition process.

3.2.1. Disjunction assumption (DJA). This assumption claims that the performance of a task cannot be achieved by more than one recipe for this task. Therefore, if a recipe for T is performed then no other of T's recipes can be achieved. To formalize this assumption, we use the following operators:
$\forall\ r \in H$ such as $r = \forall x\ B_1 \wedge \ldots \wedge B_n \supset A$ then $ant(r)= B_1 \wedge \ldots \wedge B_n$ and $csq(r)=A$.

Then the DJA set of disjunction assumptions is :

$DJA=\{\forall x\ ant(r_1) \supset \neg ant(r_2): \forall\ r_1,r_2 \in H \text{ and } csq(r_1)=csq(r_2)\}$

If a recipe r_1 for $csq(r_1)$ is (or will be) performed then no other $csq(r_1)$ recipes are achieved. In the cooking micro-world example (figure 1), this assumption claims that if we "Prepare the Pasta" and Mix With A Sauce" to "Make a Pasta Dish" then we do not "Take Cooked Pasta Dish from Freezer" or "Buy a Hot Pasta Dish" to "Make this Pasta Dish".

The following statement of DJA assumption is logically equivalent.

$DJA=\{\forall x\ \neg ant(r_1) \vee \neg ant(r_2): \forall\ r_1,r_2 \in H \text{ and } csq(r_1)=csq(r_2)\}$

In the previous example the extension of DJA set is:

$DJA=\{$ **1.** $\forall x\ \neg(B(s_1(x)) \wedge \Gamma_1) \vee \neg(C(s_2(x)) \wedge \Gamma_2);\ldots\}$

3.2.2. Exhaustion assumption (EXA) and Composition/Utilization assumption (CUA). Let EXA be the set of all statements that formalize the closed-world assumption. We thus suppose that all recipes performing the tasks are known. Furthermore, an observation can be explained by (or results) at least a known recipe.

Then the EXA statement is:

$EXA=\{\ \forall x\ csq(r) \supset ant(r) \vee ant(r_1) \vee \ldots \vee ant(r_n) : \forall\ r, r_1, \ldots, r_n \in H \text{ and } csq(r)=csq(r_i)\ \forall i \in [1,n]\ \}$

In the cooking example, the EXA set contains the following assumptions: To achieve the "Prepare Meal" task either we perform the "Make Pasta Dish" task or we carry out the "Make Meat Dish" task. Thence, the extension of EXA is:

$EXA=\{$**1.** $\forall x\ A(x) \supset (B(s_1(x)) \wedge \Gamma_1) \vee (C(s_2(x)) \wedge \Gamma_2)$;**2.** $\forall x\ B(x) \supset (X(s_2(x)) \wedge E(s_1(x)) \wedge \Gamma_3) \vee (F(s_3(x)) \wedge \Gamma_4) \vee (J(s_4(x)) \wedge \Gamma_5)$;**3.** $\forall x\ E(x) \supset (G(s_3(x)) \wedge H(s_1(x)) \wedge I(s_2(x)) \wedge \Gamma_7)$;**4.** $\forall x\ F(x) \supset (G(s_2(x)) \wedge K(s_1(x)) \wedge \Gamma_8);\ \ldots\}$

The CUA assumption states that if an observation is a part of recipes then at least one of these is used.

Then the CUA set is defined as follows:

$CUA=\{\ \forall x\ E(x) \supset (\exists y\ E_1(y) \wedge E'^{1}(s_{11}(x)) \wedge \ldots \wedge \Gamma_1 \wedge s_{1i}(y)=x) \vee (\exists y\ E_2(y) \wedge E'^{2}(s_{21}(x)) \wedge \ldots \wedge \Gamma_2 \wedge\ s_{2i}(y)=x) \vee \ldots \vee (\exists y\ E_n(y) \wedge E'^{n}(s_{n1}(x)) \wedge \ldots \wedge \Gamma_n \wedge s_{ni}(y)=x)$

Where $A=\{\forall x\ E'^{1}(s_{11}(x)) \wedge \ldots \wedge E(s_{1i}(x)) \wedge \ldots \wedge \Gamma_1 \supset E_1(x); \forall x\ E'^{2}(s_{21}(x)) \wedge \ldots \wedge E(s_{2i}(x)) \wedge \ldots \wedge \Gamma_2 \supset E_2(x);\ \ldots;\ \forall x\ E'^{n}(s_{n1}(x)) \wedge \ldots \wedge E(s_{ni}(x)) \wedge \ldots \wedge \Gamma_n \supset E_n(x)\} \subset H\}$

The A set contains all recipes owning the task type E in its antecedent. If E is observed then we can reasonably deduce (due to the closed-world assumption) that at least one recipe containing E is achieved. For example, if "Boil Water" is observed then we can infer that either "add Mash" and therefore "Hydrate Mash" are performed or "Add Pasta", "Wait" and thus "Prepare Pasta" are carried out. In this example, CUA set is:

$CUA=\{$**1.** $\forall x\ G(x) \supset\ (\exists y\ E(y) \wedge x=s_3(y) \wedge H(s_1(y)) \wedge I(s_2(y)) \wedge \Gamma_7) \vee\ (\exists y\ F(y) \wedge x=s_2(y) \wedge K(s_1(y)) \wedge \Gamma_8);\ );$ **8.** $\forall x\ F(x) \supset\ (\exists y\ B(y) \wedge x=s_3(y) \wedge \Gamma_4);$ **9.** $\forall x\ J(x) \supset\ (\exists y\ B(y) \wedge x=s_4(y) \wedge \Gamma_5);$ **10.** $\forall x\ B(x) \supset\ (\exists y\ A(y) \wedge x=s_1(y) \wedge \Gamma_1);\ \ldots\}$

Note that the CUA set only treats the observations, which are tasks. However, it is interesting to find the plans explaining a world state (as observation). Consider the CUA* set such as:

$CUA^*=CUA \cup \{\ \forall x\ \gamma \supset (A(x) \wedge B_1(s_1(x)) \wedge\ \ldots \wedge B_n(s_n(x)) \wedge \Gamma') \vee \ldots \vee (A^m(x) \wedge B^m{}_1(s^m{}_1(x)) \wedge\ \ldots \wedge B^m{}_n(s^{mn}(x)) \wedge \Gamma^{m\prime})$

Where $B=\{\forall x\ B_1(s_1(x)) \wedge B_2(s_2(x)) \wedge \ldots \wedge B_n(s_n(x)) \wedge \Gamma \supset A(x);\ \ldots;\ \forall x\ B^m1(s^m{}_1(x)) \wedge B^m2(s^m{}_2(x)) \wedge \ldots \wedge B^mn(s^m{}_n(x)) \wedge \Gamma \supset A^m(x)\ \} \subset H$ and

γ is an effects fact (thus a part of the effects constraints) belonging $\Gamma, ..,\Gamma^m$ and Γ' is the constraints formulas $\Gamma, \ldots, \Gamma^m$ without $\gamma\}$

This extension of CUA set allows us to recognize plans from an observation which is a state. Thence, for all effects predicates of all recipes the $\forall x\ \gamma \supset (A(x) \wedge B_1(s_1(x)) \wedge\ \ldots \wedge B_n(s_n(x)) \wedge \Gamma') \vee \ldots \vee (A^m(x) \wedge B^m{}_1(s^m{}_1(x)) \wedge\ \ldots \wedge B^m{}_n(s^m{}_n(x)) \wedge \Gamma^{m\prime})$ statement is added to CUA. The B set contains all H's recipes which have γ as effect.

3.2.3. Elimination assumption (ELA). This is the only assumption needing a second order statement. It claims that all eliminated trees are trees where a path between the focalized task and the observations does not exist. A tree (or plan) is formalized by a conjunction of predicates (task or constraints). To formalize this assumption, the following definitions are required.

Let Leaf(f) be the leafs set of sub-tree f such as:

$Leaf(f = \underset{j\in[1,n]}{\wedge} P_j) = \{P_j \in f : P_j \in H_T \wedge \forall f' \in [1,n]\ \forall f'\ (P_{j'} \wedge f' \supset P_j) \notin H\}$ Leaf(f) is the set of P_j such that P_j is a task type (thus an element of H_T) belonging to f and for all $P_{j'}$ of f , a recipe for P_j does not exist in f.

Root(f) is the root set of sub-tree f such as:

$Root(f = \underset{j\in[1,n]}{\wedge} P_j) = \{P_j \in f : P_j \in H_T \wedge \forall f' \in [1,n]\ \forall f'\ (P_j \wedge f' \supset P_{j'}) \notin H\}$ Root(f) is the set of P_j such as P_j is a task type belonging to f and for all $P_{j'}$ of f , a recipe where P_j is a subtask does not exist.

Let Ftree(f) be a predicate which is true if f is a Full Tree. A formula is a Full Tree if and only if all f's roots are roots of H or a focalized task and all f's leaves are leaves of H or observations.

$$FTree(f = \underset{j\in[1,n]}{\wedge} P_j) = \forall j \in [1,n]\ (P_j \in Leaf(f) \supset P_j \in Obs \vee P_j \in H_E) \vee (P_j \in Root(f) \supset P_j \in Foc \vee P_j \in H_S)$$

Way(f) defines all f's paths such as:

$$Way(f = \underset{j\in[1,n]}{\wedge} P_j') = \text{The set of all paths}$$
$$\{ \underset{i\in[1,m]}{\wedge} P_i \ :\ \text{Such that one root exists and}\ \exists i \in [1,m]\ ((P_i \in Root(f)) \wedge \text{All elements of this subset are linked by a recipe}\ \forall j \in [1,m]\ (i \neq j \supset [(P_j \notin Root(f)) \wedge \exists j' \in [1,m]\ (\forall f'\ (P_j \wedge f' \supset P_{j'}) \in H)]))\}$$

When a plan recognition process is applied, often several plans can explain the observations. As a plan is represented by a predicate conjunction (task type and constraints) then each plan (which can explain observations) is associated by the $\vee$ (or) connector.

The ELA assumption consists of a statement of the following form:

$$\begin{aligned}
&\textit{If}\quad f \text{ represents the candidate plans (predicate disjunctions of conjunctions)}\\
&\quad \forall f\,(f = \underset{i\in[1,m]}{\vee} \underset{j\in[1,n]}{\wedge} P_j^i) \wedge (\forall i \in [1,m]\ (\forall j \in [1,n]\ \text{Predicate}\ (P_j^i))) \wedge\\
&\quad \text{a sub-tree (a plan) exist}\\
&\quad [(\exists i_1 \in [1,m]\\
&\quad \text{which is full tree } (f') \text{ and}\\
&\quad (FTree\ (f' = \underset{j\in[1,n]}{\wedge} P_j^{i_1}) \wedge\\
&\quad \text{that does not contains observations}\\
&\quad \text{or a focalized task or}\\
&\quad ((\forall j \in [1,n]\ P_j^{i_1} \notin Obs \vee P_j^{i_1} \notin Foc) \vee\\
&\quad \text{two tasks exist which are either an}\\
&\quad \text{observation and a focalized task}\\
&\quad \text{(or vice-versa) and}\\
&\quad (\exists j, \forall j' \in [1,n]\ ((P_j^{i_1} \in Obs \wedge P_{j'}^{i_1} \in Foc) \vee (P_j^{i_1} \in Foc \wedge P_{j'}^{i_1} \in Obs)) \wedge\\
&\quad \text{whatever the } f'\text{'s ways there is no}\\
&\quad \text{bind between these two tasks}\\
&\quad \forall w \in Way\ (f')\ \ w = \underset{l\in[1,k]}{\wedge} P_j' \wedge\\
&\quad (\forall l, l' \in [1,k]\ \neg(P_l' = P_j^{i_1}) \vee \neg(P_{l'}' = P_{j'}^{i_1})))]\\
&\textit{Then}\quad \text{this sub-tree is eliminated}\\
&\quad \supset (\underset{\substack{i\in[1,m]\\ i\neq i_1}}{\vee} \underset{j\in[1,n]}{\wedge} P_j^{'i}) \vee F
\end{aligned}$$

The ELA assumption allows us to eliminate the candidate trees

(or plans) which do not explain the observations from the focalized task. In the figure 1 example, suppose that a task of type "Prepare Meal" and "Make Pasta Dish" constitute a plan (or tree) and "Prepare Meal" and "Make Meat Dish" represent another plan (or tree). These two candidate plans are translated by the formulas $\forall x\ (A(x) \wedge B(s_1(x))) \vee (A(x) \wedge B(s_2(x)))$. Suppose now, that the focalized task is a task of type "Prepare Meal" and the observation is "Wait", then the ELA assumption eliminates the second plan to obtain $\forall x\ (A(x) \wedge B(s_1(x))) \vee F$ which is logically equivalent to $\forall x\ (A(x) \wedge B(s_1(x)))$.

3.3. Keyhole plan recognition inferences

In our theory, plans are recognized from the following premises: H ∪ Plan Recognition Assumptions ∪ Obs ∪ Foc, where Plan Recognition Assumptions are some above assumptions and Obs is the formula containing the observations and Foc is the focalized task. Then

H ∪ Plan Recognition Assumptions ∪ Obs ∪ Foc $\models$ f

with $f = \bigvee_{i\in[1,m]} \bigwedge_{j\in[1,n]} P_j^i$ or F

If the premises are inconsistent (entails F, $\models$ F) then no plan can explain the observations Obs from the focalized task Foc with the H's task type hierarchy and the considered Plan Recognition Assumptions. Otherwise, if they are consistent then they entail a formula f, which represents the disjunction of candidates plans (or trees).

4. Different strategies of plan recognition

In the plan recognition scenarios presented below, we suppose that the recipe's constraints are always consistent. Of course, this is usually not the case, but constraint management is not required to illustrate our subject.

4.1. Bottom-Up Recognition

This kind of recognition starts with the observations and goes up to the focalized task. The Plan Recognition Assumptions required are CUA*, DJA, ELA. In the example of the cooking world, suppose that the task u of type G is observed and the focalized task is z of type A.

Then the Bottom-up recognition premises are: H ∪ CUA* ∪ DJA ∪ ELA ∪ {G(u)} ∪ {A(z)}. We choose only to indicate the assumptions used (by their name and number).

First, ELA do nothing, and then

CUA(1) $\models (A(z) \wedge G(u) \wedge E(*i_1) \wedge u=s_3(*i_1) \wedge H(s_1(*i_1)) \wedge$
$I(s_2(*i_1)) \wedge \Gamma_7) \vee\ (A(z) \wedge G(u) \wedge F(*i_2) \wedge$
$u=s_2(*i_2) \wedge K(s_1(*i_2)) \wedge \Gamma_8)$

To alleviate the notation the constant prefixed with a "*" stands in place of existentially quantified variables.

Then, ELA entails no change, CUA (8) and CUA (6) are applied (we thus choose to develop the "father" tasks)

CUA (8) $\models [\ A(z) \wedge G(u) \wedge E(*i_1) \wedge u=s_3(*i_1) \wedge H(s_1(*i_1))$

CUA (9) $\wedge I(s_2(*i_1)) \wedge \Gamma_7 \wedge \underline{B(*i_1') \wedge X(s_2(*i_1')) \wedge\ *i_1=s_1(*i_1') \wedge \Gamma_3}\ ] \vee [\ A(z) \wedge G(u) \wedge F(*i_2) \wedge$
$u=s_2(*i_2) \wedge K(s_1(*i_2)) \wedge \Gamma_8 \wedge \underline{B(*i_2') \wedge i_2=s_3(*i_2')}$
$\underline{\wedge \Gamma_4}\ ]$

As in the precedent step, CUA (10) is now applied on the two sub-tree (or plans)

CUA (10) $\models [\ A(z) \wedge G(u) \wedge E(*i_1) \wedge u=s_3(*i_1) \wedge H(s_1(*i_1)) \wedge I(s_2(*i_1)) \wedge \Gamma_7 \wedge B(*i_1') \wedge X(s_2(*i_1')) \wedge\ *i_1=s_1(*i_1') \wedge \Gamma_3$
$\wedge\ \underline{A(*i_1'') \wedge *i_1'=s_1(*i_1'') \wedge \Gamma_1}\ ] \vee [\ A(z) \wedge G(u) \wedge F(*i_2) \wedge u=s_2(*i_2) \wedge K(s_1(*i_2)) \wedge \Gamma_8 \wedge B(*i_2') \wedge *i_2=s_3(*i_2') \wedge$
$\Gamma_4 \wedge \underline{A(*i_2'') \wedge *i_2'=s_1(*i_2'') \wedge \Gamma_1}]$

Suppose that z can be unify to $*i_1''$ and $*i_2''$, and after the simplifications undertaken by the equality predicate. The following result is obtained:

$\models [\ A(z) \wedge G(u) \wedge u=s_3(s_1(s_1(z))) \wedge E(s_1(s_1(z))) \wedge$
$H(s_1(s_1(s_1(z)))) \wedge I(s_2(s_1(s_1(z)))) \wedge B(s_1(z)) \wedge$
$X(s_2(s_1(z))) \wedge \Gamma_3\ \wedge \Gamma_1 \wedge \Gamma_7\ ] \vee [\ A(z) \wedge G(u) \wedge$
$u=s_2(s_3(s_1(z))) \wedge F(s_3(s_1(z))) \wedge K(s_1(s_3(s_1(z)))) \wedge$
$B(s_1(z)) \wedge \Gamma_4\ \wedge \Gamma_1 \wedge \Gamma_8\ ]$

4.2. Top-Down Recognition

Top-Down recognition starts from the focalized task and goes down to reach the observed tasks. This kind of recognition needs the DJA, EXA, ELA Plan Recognition Assumptions. Consider the following scenario in the micro-cooking world previously described. The focalized task and the observed task remain the same (z of type A and u of type W).
ELA is applied but does not provoke further conclusions, therefore EXA (1) is applied
EXA (1) $\models [\ A(z) \wedge G(u) \wedge \underline{B(s_1(z)) \wedge \Gamma_1}\ ] \vee [\ A(z) \wedge G(u) \wedge \underline{C(s_2(z)) \wedge \Gamma_2}\ ]$

The application of ELA assumption result in the elimination of the second plan because it is a Full Tree and no path exists between A(z) and G(u). Then we obtain:
ELA $\models A(z) \wedge G(u) \wedge B(s_1(z)) \wedge \Gamma_1$
Then we develop $s_1(z)$ task with EXA (2)
EXA (2) $\models [\ A(z) \wedge G(u) \wedge B(s_1(z)) \wedge \Gamma_1 \wedge \underline{X(s_2(s_1(z))) \wedge E(s_1(s_1(z))) \wedge \Gamma_3}\ ] \vee [\ A(z) \wedge G(u) \wedge B(s_1(z)) \wedge \Gamma_1 \wedge \underline{F(s_3(s_1(z))) \wedge \Gamma_4}\ ] \vee [A(z) \wedge G(u) \wedge B(s_1(z)) \wedge \Gamma_1 \wedge \underline{J(s_4(s_1(z))) \wedge \Gamma_5}\ ]$

The ELA eliminates the third plan (for the previous reasons)
ELA $\models [\ A(z) \wedge G(u) \wedge B(s_1(z)) \wedge \Gamma_1 \wedge X(s_2(s_1(z))) \wedge E(s_1(s_1(z))) \wedge \Gamma_3\ ] \vee [\ A(z) \wedge G(u) \wedge B(s_1(z)) \wedge \Gamma_1 \wedge F(s_3(s_1(z))) \wedge \Gamma_4\ ]$

The EXA (3) and EXA (4) is now applied to develop $s_1(s_1(z))$ and $s_3(s_1(z))$ tasks.
EXA (3) $\models [\ A(z) \wedge G(u) \wedge B(s_1(z)) \wedge \Gamma_1 \wedge X(s_2(s_1(z))) \wedge$ EXA(4) $E(s_1(s_1(z))) \wedge \Gamma_3 \wedge \underline{G(s_3(s_1(s_1(z))))} \wedge \underline{H(s_1(s_1(s_1(z)))) \wedge I(s_2(s_1(s_1(z)))) \wedge \Gamma_7}] \vee [\ A(z) \wedge G(u) \wedge B(s_1(z)) \wedge \Gamma_1 \wedge F(s_3(s_1(z))) \wedge \Gamma_4 \wedge \underline{G(s_2(s_3(s_1(z))))} \wedge \underline{K(s_1(s_3(s_1(z)))) \wedge \Gamma_8}\ ]$

Suppose that we can unify u to $s_3(s_1(s_1(z)))$ and $s_2(s_3(s_1(z)))$, Then these two plans are recognized:
$\models [\ A(z) \wedge G(u) \wedge u= s_3(s_1(s_1(z)))) \wedge B(s_1(z)) \wedge X(s_2(s_1(z))) \wedge E(s_1(s_1(z))) \wedge H(s_1(s_1(s_1(z)))) \wedge I(s_2(s_1(s_1(z)))) \wedge \Gamma_7 \wedge \Gamma_1 \wedge \Gamma_3\ ]$
$\vee [\ A(z) \wedge G(u) \wedge u=s_2(s_3(s_1(z)))) \wedge B(s_1(z)) \wedge F(s_3(s_1(z))) \wedge K(s_1(s_3(s_1(z)))) \wedge \Gamma_8 \wedge \Gamma_1 \wedge \Gamma_4\ ]$

This outcome is identical to the outcome of the Bottom-Up recognition.
Suppose now, we observe a new task e of type I, then this observation is added to the premises.
I(e) $\models [\ A(z) \wedge G(u) \wedge u= s_3(s_1(s_1(z)))) \wedge B(s_1(z)) \wedge X(s_2(s_1(z))) \wedge E(s_1(s_1(z))) \wedge H(s_1(s_1(s_1(z)))) \wedge I(s_2(s_1(s_1(z)))) \wedge \Gamma_7 \wedge \Gamma_1 \wedge \Gamma_3 \wedge \underline{I(e)}\ ] \vee [\ A(z) \wedge G(u) \wedge u=s_2(s_3(s_1(z)))) \wedge B(s_1(z)) \wedge F(s_3(s_1(z))) \wedge K(s_1(s_3(s_1(z)))) \wedge \Gamma_8 \wedge \Gamma_1 \wedge \Gamma_4 \wedge \underline{I(e)}\ ]$

Then the ELA is applied, and the second plan is eliminated because it is a Full Plan and no path exists between A(z) and I(u). However, we consider that we can unify e to $s_2(s_1(s_1(z)))$ then the first plan is not eliminated. We thus obtain:
$\models A(z) \wedge G(u) \wedge I(e) \wedge e= s_2(s_1(s_1(z))) \wedge u= s_3(s_1(s_1(z)))) \wedge$
$B(s_1(z)) \wedge X(s_2(s_1(z))) \wedge E(s_1(s_1(z))) \wedge H(s_1(s_1(s_1(z)))) \wedge$
$I(s_2(s_1(s_1(z)))) \wedge \Gamma_7 \wedge \Gamma_1 \wedge \Gamma_3$

5. Algorithm

A Top-Down plan recognition algorithm has been implemented and only works with the conjunction of observations. The observations are treated one after another. First we look for the plans explaining one observation from the focalized task, and then the same work is repeated with another observation and so on. When a new observation is treated, we only keep the plans containing (or entailing) it and the older observations. The way of recognizing the observation conjunction is illustrated by the Top-Down scenario studied above.

The ELA assumption can be implemented with a hash table. This hash table is indexed (key words) with the tasks of the T-Graph (Task Graph containing the different candidate plans) under construction. The possible values of this hash table are "U for unknown", "T for true" and "F for false". The default value for all tasks is U, when an observation is found then all father tasks in the T-Graph are set to T. An elementary task, which is not, an observation is set to F. If all subtasks of a recipe in a T-Graph are F then the recipe is eliminated. This algorithm can also recognize the world state observations.

The implemented algorithm has a polynomial complexity. Consider $n= |H_T| + \varepsilon$, where $|H_T|$ is the number of task types and ε the number of recipes containing the same task type. Thence, the algorithm complexity (worst case) is $O(n^4)$. Of course, these complexities do not include the checking time for constraints. However, this calculation of complexity supposes (worst

case) that all recipes are applicable, but in a real system this case cannot occur. The algorithm proposed can be implemented in a generic way, i.e. the constraints checking and therefore the recipes for the analyzed task can be an input of the plan recognition process. A generic version of this algorithm is implemented in Java.

6. Related work

The theory and formalization presented here is very similar to the Kautz's logical theory of keyhole plan recognition [9,10,11]. Nevertheless, Kautz's event type hierarchy uses two relationships between events (whereas in our theory, the events can be view as tasks). The *"has part"* relationship corresponds to our *"composed of"* relation described inside the recipe notion. While, the *"is a"* relation is not directly represented in our task type hierarchy. We think the *"is a"* relationship can be translated by the "a way to achieve" relationship. This relationship can be viewed as one means of naming a recipe. However, the *"is a"* abstraction relationship increases the difficulty of implementation. Furthermore, the algorithm complexity is exponential in the size of event hierarchy.

Kautz also formalizes the basic step of recognition plan reasoning, but he treats (and thus formalizes) only the bottom-up recognition which limits its application to the focalized discourse understanding. Moreover, in his theory only the event observations are considered, it is impossible to recognize plans from a world state observation.

Lesh in [12] proposes a fast plan recognizer, which is used to find the user's plans in a UNIX command world. To increase the performance, all plans explaining all possible observations are pre-established. Therefore, the recognition process is reduced to a traverse of pre-established plans. Of course, the plan pre-establishment is very expensive (in time and memory) and therefore impossible in several applications, but is only constructed once. However, we believe that the pre-establishment of plans for limiting the recognition process to can be adapted to some applications. Moreover, this strategy of keyhole plan recognition can be represented in our formalization.

7. Conclusion and future works

The presented logical formalization of keyhole plan recognition gives a synthetic and useful vision of plan recognizer. This description allows us to precisely and consistently specify the recognizer framework. The explicit dissociation between the specific plan recognition aspects and other aspects allows us to adapt a plan recognition strategy and algorithm to different application contexts. Moreover, the Exhaustion and Composition/Utilization assumptions provide two basic ways to traverse a plan, thus they allow to formalized a large number of plan recognition strategies.

A Top-Down plan recognition process is also proposed, and seems well suited to the discourse plan-based approaches theories [7,8,14,15,16]. The Top-Down recognition implemented in Java language has a $O(n^4)$ complexity (n being the number of task types plus the number of recipes containing the same task types). This algorithm can run in parallel, which increases its performance by decreasing the n value. We will then distribute the Top-Down algorithm (to run it in parallel) on several machines by using the CORBA/IDL technology. Furthermore, this algorithm will be a part of the communication manager in a cooperative knowledge based system (developed by our team [19]), and will be applied to a cooperative help system for aircraft maintenance.

7. References

[1] J.F. Allen. Towards a general theory of action and time. In Artificial Intelligence, 23:123-154, 1984.

[2] J.F. Allen, B. Miller, E. Ringger, and T. Sikorski. A Robust System for Natural Spoken Dialogue. In *Proc. 34th Meeting of the Assoc. for Computational Linguistics,* 1996.

[3] P.R. Cohen, and H.J. Levesque, Persistence, Intention, and Commitment. In *P.R Cohen, J.L. Morgan, M.E. Pollack, editors, Intention in Communication.* MIT Press, Cambridge, MA, 1990.

[4] G. Ferguson, and J.F. Allen. Generic Plan Recognition for Dialogue Systems. In *ARPA Workshop on Human Language Technology,* Princeton, NJ, pp 21-23 March, 1993.

[5] G. Ferguson, and J.F. Allen TRIPS. An Integrated Intelligent Problem-Solving Assistant, *In* Proceedings of the Fifteenth National Conference on AI (AAAI-98)*,* Madison, WI, 26--30 July, 1998.

[6] B.J. Grosz, and S. Kraus. Collaborative plans for complex group actions. In *Artificial Intelligence,* 86(2), pp. 381–386., October 1996.

[7] B.J. Grosz, and C.L. Sidner. Attention, intentions, and the structure of discourse. In *Computational Linguistics, 12(3),* pp 175-204, 1986.

[8] B.J. Grosz, and C.L. Sidner. Plans for discourse. In *P.R Cohen, J.L. Morgan, M.E. Pollack, editors, Intention in Communication.* MIT Press, Cambridge, MA, pp 417-444, 1990.

[9] H.A. Kautz. A Formal Theory Of Plan Recognition. PhD thesis, University of Rochester, 1987.

[10] H.A. Kautz. A formal theory of plan recognition and its implementation. In *Reasoning about Plans,* pp 69-126, Morgan Kaufmann, 1991.

[11] H.A. Kautz, and J.F. Allen. Generalized plan recognition. In *Proc 5th Nat. Conf. AI*, pp 32-37, 1986.

[12] N. Lesh, and O. Etzioni. A sound and fast goal recognizer. In *Proc. 14th Int Joint Conf. AI*, pp 1704-1710, 1995.

[13] N. Lesh, C. Rich, and C.L. Sidner. Using Plan Recognition in Human-Computer Collaboration. MERL Technical Report 98-23 December 1998, In *Seventh Int. Conf. on User Modeling*, Banff, Canada, June 1999.

[14] D.J. Litman, and J.F. Allen. A plan recognition model for subdialogues in conversations. In *Cognitive Science* 11: pp 163-200, 1987.

[15] K.E. Lochbaum. Using collaborative plans to model the intentional structure of discourse. IN *Technical Report TR-25-94, Harvard Univ, Ctr. for Res. In Computing Tech.* PhD thesis, 1994.

[16] K.E. Lochbaum. A collaborative planning model of intentional structure. In *Computational Linguistics,* Volume 24, Number 4, pp 525-572, 1998.

[17] M.E. Pollack. Plans as Complex Mental Attitudes. In *P.R Cohen, J. Morgan, and M.E Pollack, editors, Intentions in Communication.* MIT Press, 1990.

[18] C. Rich, and C.L. Sidner. COLLAGEN: A collaboration manager for software interface agents. In *User Modeling and User-Adapted Interaction, 1998*. Special issue on Computational Models of Mixed-Initiative Interaction. Forthcoming. Also published as MERL Technical Report 97-21a, 1998.

[19] JL. Soubie. Coopération et système à base de connaissances. *Mémoire d'habilitation, UPS Toulouse III,* IRIT, France, 1996.

[20] A. Waern.. Recognising human plans: Plan recognition in human-computer interaction. PhD diss, Swedish Institute of Computer Science, 1996.