

Intention recognition in the Situation Calculus and Probability Theory frameworks

Robert Demolombe¹ and Ana Mara Otermin Fernandez² *

¹ ONERA Toulouse

France

Robert.Demolombe@cert.fr

² ONERA Toulouse

France

txantxita@hotmail.com

Abstract. A method to recognize agent's intentions is presented in a framework that combines the logic of Situation Calculus and Probability Theory. The method is restricted to contexts where the agent only performs procedures in a given library of procedures, and where the system that intends to recognize the agent's intentions has a complete knowledge of the actions performed by the agent.

An original aspect is that the procedures are defined for human agents and not for artificial agents. The consequence is that the procedures may offer the possibility to do any kind of actions between two given actions, and they also may forbid to perform some specific actions. Then, the problem is different and more complex than the standard problem of plan recognition.

To select the procedures that partially match the observations we consider the procedures that have the greatest estimated probability. This estimation is based on the application of Bayes' theorem and on specific heuristics. These heuristics depend on the history and not just on the last observation.

A PROLOG prototype of the presented method has been implemented.

1 Introduction

When two agents have to interact it is important for each agent to know the other agent's intentions because this knowledge allows to anticipate his future behavior. This information can be used either to help the other agent to do what he intends to do or to control whether what he does is compatible with his intention. Even if an agent can never be sure that he knows the other agent's intentions an uncertain information is much better than a complete ignorance when a decision has to be taken.

In this paper a method is proposed to recognize what are the agent's intentions in the particular context of a pilot that interacts with an aircraft. The first specificity of this context is that the pilot performs procedures that are very

* Also student at: Universidad Politenica de Madrid.

well defined in a handbook. The second specificity is that the procedures are defined in terms of commands that have to be performed (like to turn a switch on) and it is reasonable to assume that the performance of these commands can be perceived thanks to sensors in the aircraft. Then, it is possible to design a system (for instance a part of the automatic pilot of the aircraft) that has the capacity to observe all the commands performed by the pilot.

Under this assumption the system can compare the sequence of observations with the procedure definitions in the handbook and it can determine the procedures that match with these observations. The procedures that have the “best” match are assigned to the agent’s intentions.

To define a method to recognize the pilot’s intentions we have to find solutions to three independent problems:

1. to select a language to represent the procedures in formal terms,
2. to define a formal characterization of the procedures that match with the observations,
3. to define a method to select the procedures that have the “best” match and are assigned to the agent’s intention.

In a previous work Demolombe and Hamon [6, 10] have proposed solutions to problems 1 and 2 in the logical framework of the Situation Calculus. The Situation Calculus is a variant of classical first order logic, that is the reason why it is more convenient for computational logic than modal logics.

The contribution of this paper is to propose a solution to problem 3 in a framework that combines Situation Calculus and Probability Theory and which is based on Bayes’ theorem. Probabilities have already been used in combination with Situation Calculus in [12] to deal with no deterministic actions, but that is a quite different problem.

There are many other works that have similar objectives in the field of plan recognition [13] and many of them make use of probabilities [4, 8, 1, 5] or use an utility function [15]. Baier in [3] also uses the framework of the Situation Calculus but without probabilities. Many of them have been designed in the particular context of natural language analysis [7, 2, 5] or game theory [1].

The original feature in our case is that the pilot’s procedures may allow any other command in between a sequence of two prescribed commands and it may be specified that some commands are forbidden. Also it may happen that the pilot has the intention to perform several procedures in parallel. The consequence is that problems 2 and 3 are much more complex than the standard problem of plan recognition.

The paper is organized as follows. In sections 2 and 3 the solutions to problems 1 and 2 are recalled. In section 4 the method to solve problem 3 is presented. In that section we start with the analysis of a typical example, we define a general method to compute probabilities, we define heuristics to estimate the probabilities and finally we apply the method to the example to show that the results fit the intuitive requirements. Possible refinements or extensions of the method are presented in the conclusion.

Since the method can be applied to many other contexts we shall use the general term “agent” instead of “pilot”, and “action” instead of “command”.

2 A brief introduction to the Situation Calculus and to a GOLOG extension

The logical framework of Situation Calculus [17] is used to represent the states of the world and the actions that are performed by the agent.

The Situation Calculus is a typed first order logic with equality (except some limited fragments that are of second order). In the language there are two kinds of predicates. The predicates whose truth value may change after performance of an action are called “fluents”. They have exactly 1 argument of the type situation which is the last argument. The other predicates have no argument of the type situation.

For example, we may have the predicates:

nationality(x): the nationality of the aircraft is x .

gear.extended(s): in the situation s the landing gear is extended.

altitude(x, s): in the situation s the aircraft altitude is x .

Here *altitude(x, s)* and *gear.extended(s)* are fluents, and *nationality(x)* is not a fluent.

The terms of type situation may be constant or variable symbols of the type situation, or terms of the form $do(a, s)$ where do is a designated function symbol, a is a term of type action and s is a term of type situation.

For instance, if S_0 is a constant of type situation and *extend.gear* and *retract.gear* are constants of type action, the following terms are of type situation: S_0 , $do(extend.gear, S_0)$, $do(extend.gear, s)$ and $do(retract.gear, do(extend.gear, S_0))$.

The term $do(retract.gear, do(extend.gear, S_0))$ denotes the situation where we are after performance of the actions *extend.gear* and *retract.gear*.

As a matter of simplification we use the notation $do([a_1, \dots, a_n], s)$ to denote $do(a_n, \dots, do(a_1, s) \dots)$.

The grammar of the formulas of the Situation Calculus is defined as usual for classical first order logics.

A successor relation ³ is defined on the set of situations. Intuitively $s \leq s'$ means that the situation s' is reached from the situation s after some sequence of action. In semiformal terms, $s \leq s'$ is the smallest relation that satisfies the following properties:

$$\begin{aligned} s \leq s' &\stackrel{\text{def}}{=} (s < s') \vee (s = s') \\ \forall s \forall s' \forall a (s' = do(a, s) &\rightarrow s < s') \\ \forall s \forall s' \forall s'' ((s < s') \wedge (s' < s'')) &\rightarrow (s < s'') \end{aligned}$$

To define the truth value of the fluents in any situation a successor state axiom has to be given for each fluent. For example, for *gear.extended(s)* we have:

³ In this paper the definition of the successor relation is the only part of the Situation Calculus that requires second order logic.

$$\forall s \forall a (\text{gear.extended}(\text{do}(a, s)) \leftrightarrow a = \text{extend.gear} \vee \text{gear.extended}(s) \wedge \neg(a = \text{retract.gear}))$$

The intuitive meaning of this axiom is that the only action that can cause $\text{gear.extended}(\text{do}(a, s))$ to be true (resp. false) is the action extend.gear (resp. retract.gear).

The GOLOG language [14] is a programming language for robots but it can be used for other kinds of agents. Its expressive power is the same as ALGOL and its semantics is defined in the logic of the Situation Calculus. Programs are terms that represent complex actions defined with several operators.

Here, for simplicity, we have only considered the operator of sequence (denoted by “;”), test (denoted by “ $\phi?$ ”) and non deterministic choice (denoted by “|”). To represent what is called in the following “procedures” we have added the “negation” operator (denoted by “-”) and the “any sequence of actions” term (denoted by “ σ ”). The motivation of this extension can be explained with the following example.

Let us, consider the procedure called “fire on board”, which is described for a small private aircraft. The procedure says that in case of engine fire the pilot 1) turns off fuel feed, 2) sets full throttle, and 3) sets mixture off. These three primitive actions, or commands, are respectively denoted by fuel.off , full.throttle and mixture.off , and the procedure is denoted by fire.on.board .

However, it is implicit in the procedure definition that between actions 1) and 2) or between 2) and 3) the pilot can do any other action. For example, he can call air traffic control. It is also implicit that after turning off fuel feed he must not turn on fuel feed. That is just common sense for a human being but it has to be made explicit to define a formal method that can be used by the system which observes the pilot.

Then, in the modified GOLOG language the “fire on board” procedure is represented by:

$$\text{fire.on.board} \stackrel{\text{def}}{=} \text{fuel.off}; (\sigma / \text{fuel.on}); \text{full.throttle}; (\sigma / \text{fuel.on}); \text{mixture.off}$$

where α_1 / α_2 is an abbreviation for $\alpha_1 - (\sigma; \alpha_2; \sigma)$ which intuitively means that the sequence of actions which is a performance of α_1 must not contain a sequence of actions which is a performance of α_2 .

In the case of programs for an artificial agent there is no need for the term σ nor for the operator “/” because **an artificial agent only does what is specified in the program**. That makes the basic difference between a program and what is called here a “procedure”.

The formal definition of the modified GOLOG language is :

- atomic actions, test actions and σ are procedures,
- if α_1 and α_2 are procedures, then $(\alpha_1; \alpha_2)$, $(\alpha_1 | \alpha_2)$ and $(\alpha_1 - \alpha_2)$ are procedures.

The formal definition of the procedures is defined by formulas of the Situation Calculus language. These formulas are denoted by the property $Do_p(\alpha, s, s')$ whose intuitive meaning is:

$Do_p(\alpha, s, s')$: s' is a situation that can be reached from the situation s after performance of the procedure α .

The formal semantics of $Do_p(\alpha, s, s')$ is:

$$Do_p(a, s, s') \stackrel{\text{def}}{=} s' = do(a, s) \text{ if } a \text{ is an atomic action.}$$

$$Do_p(\sigma, s, s') \stackrel{\text{def}}{=} s \leq s'$$

$$Do_p(\phi?, s, s') \stackrel{\text{def}}{=} \phi[s] \wedge s' = s$$

$$Do_p(\alpha_1; \alpha_2, s, s') \stackrel{\text{def}}{=} \exists s_1 (Do_p(\alpha_1, s, s_1) \wedge Do_p(\alpha_2, s_1, s'))$$

$$Do_p(\alpha_1 | \alpha_2, s, s') \stackrel{\text{def}}{=} Do_p(\alpha_1, s, s') \vee Do_p(\alpha_2, s, s')$$

$$Do_p(\alpha_1 - \alpha_2, s, s') \stackrel{\text{def}}{=} Do_p(\alpha_1, s, s') \wedge \neg Do_p(\alpha_2, s, s')$$

This modified GOLOG language gives a solution to the problem 1 that we have mentioned in the introduction.

3 Doing a procedure

To characterize the fact that a sequence of performed actions “matches” a partial performance of a procedure, in the sense that this sequence **can be interpreted** as a partial performance of the procedure, we use the property $Doing(\alpha, s, s')$. However, this property does not guarantee that the agent is performing this procedure.

In informal terms the property $Doing(\alpha, s, s')$ holds if the three following conditions are satisfied:

1. The agent has begun executing a part α' of α between s and s' .
2. The agent has not completely executed α between s and s' .
3. The actions performed between s and s' do not prevent the continuation of the execution of α .

In a first step we define the property $Dom(\alpha, s, s')$ whose intuitive meaning is that we have $Do_p(\alpha, s, s')$ and there is no shorter sequence of actions between s and s' such that we have Do_p for this sequence. We have:

$$Dom(\alpha, s, s') \stackrel{\text{def}}{=} Do_p(\alpha, s, s') \wedge \neg \exists s_1 (Do_p(\alpha, s, s_1) \wedge s_1 \leq s')$$

Then, we define the property $Do_s(\alpha, s, s')$ whose intuitive meaning is that the sequence of actions between s and s' satisfies the above conditions 1, 2 and 3. We have:

$$Do_s(\alpha, s, s') \stackrel{\text{def}}{=} \exists \alpha' (start(\alpha', \alpha) \wedge \exists s_1 (s_1 \leq s' \wedge Dom(\alpha', s, s_1)) \wedge \neg \exists s_2 (s_2 \leq s' \wedge Do_m(\alpha, s, s_2)) \wedge \exists s_3 (s' < s_3 \wedge Do_m(\alpha, s, s_3)))$$

where $start(\alpha', \alpha)$ means that α can be reformulated into a procedure of the form: $(\alpha'; \alpha'') | \beta$ which has the same semantics as α , i.e. $\forall s \forall s' (Do_p(\alpha, s, s') \leftrightarrow Do_p((\alpha'; \alpha'') | \beta, s, s'))$.

The condition 1 is expressed by $\exists \alpha' (start(\alpha', \alpha) \wedge \exists s_1 (s_1 \leq s' \wedge Dom(\alpha', s, s_1)))$, the strict interpretation of condition 2 is expressed by $\neg \exists s_2 (s_2 \leq s' \wedge Do_m(\alpha, s, s_2))$, and the condition 3 is expressed by $\exists s_3 (s' < s_3 \wedge Do_m(\alpha, s, s_3))$.

Finally, the definition of $Doing(\alpha, s, s')$ is:

$$Doing(\alpha, s, s') \stackrel{\text{def}}{=} \exists s_1 (s \leq s_1 \wedge Do_s(\alpha, s_1, s')) \wedge \neg \exists s_2 (s \leq s_2 \wedge s_2 < s_1 \wedge Do_s(\alpha, s_2, s_1))$$

The condition $\exists s_1 (s \leq s_1 \wedge Do_s(\alpha, s_1, s'))$ expresses that there is an execution of α that has begun in s_1 and has not ended, and the condition $\neg \exists s_2 (s \leq s_2 \wedge s_2 < s_1 \wedge Do_s(\alpha, s_2, s_1))$ expresses that there is no previous α execution which has started and not ended before s_1 .

4 Intention recognition

This section presents a method for choosing between several procedures, that satisfy the *Doing* property, the one that can be assigned by the system to the agent's intention.

This assignment is never guaranteed to correspond to the true agent's intention, and due to this uncertainty it is sensible to make use of probabilities to make the choice.

Before going into the formal presentation of the method let us give a simple example to intuitively show what are the basic guidelines⁴ and assumptions of the method.

4.1 A simple example

Let us consider the three following procedures⁵.

$$\alpha = a; \sigma; b; \sigma; c$$

$$\beta = d; \sigma; e$$

$$\gamma = a; \sigma; f$$

Let us assume that we are in the situation s_5 where the following sequence of actions has been performed: $[f, a, d, b, c]$, that is in formal terms:

$$s_5 = do([f, a, d, b, c], s_0).$$

In the situation $s_1 = do(f, s_0)$ there is no procedure which is compatible with the performed action f . We have $\neg Doing(\alpha, s_0, s_1)$, $\neg Doing(\beta, s_0, s_1)$ and $\neg Doing(\gamma, s_0, s_1)$.

We have adopted the following assumption.

Assumption H1. If an agent has the intention to do a procedure α then he does the actions that are defined by the procedure α .

According to H1 in s_1 the system knows that the agent did not have the intention to do α in s_0 , because if he had the intention to do α in s_0 he would have started to do α and he would have done the action a in s_1 instead of f . The same for β and γ .

⁴ These guidelines are expected properties and they should not be confused with the assumptions.

⁵ In previous sections α , β and γ are procedure variables. Here specific constants are assigned to these variables.

Nevertheless in s_0 the system can accept that the probability that the agent has the intention to do α is not equal to 0. Then, we have accepted the additional assumption:

Assumption H2. If the agent in the situation s_i is not doing α , in the sense that $\neg Doing(\alpha, s_0, s_i)$, then in s_i the probability that he has the intention to do α is independent of s_i , and this probability is denoted by $\pi(\alpha)$.

Let us define the following notations.

$P(\phi)$: probability that ϕ holds.

$Int(\alpha, s_i)$: in the situation s_i the agent has the intention to do α ⁶.

In formal terms H2 can be expressed by:

$$\forall s \forall s' \forall \alpha (s \leq s' \wedge \neg Doing(\alpha, s, s') \rightarrow P(Int(\alpha, s')) = \pi(\alpha))$$

Since for any procedure α we have $\neg Doing(\alpha, s_0, s_0)$, from H2 we have: $P(Int(\alpha, s_0)) = P(Int(\alpha, s_1)) = \pi(\alpha)$, $P(Int(\beta, s_0)) = P(Int(\beta, s_1)) = \pi(\beta)$ and $P(Int(\gamma, s_0)) = P(Int(\gamma, s_1)) = \pi(\gamma)$.

In the situation $s_2 = do([f, a], s_0)$ we have $\neg Doing(\beta, s_0, s_2)$ and $P(Int(\beta, s_2)) = \pi(\beta)$, and now we have $Doing(\alpha, s_0, s_2)$ and $Doing(\gamma, s_0, s_2)$.

The fact that the action a has been performed is a good argument for the system to believe that the agent has the intention to do α and to believe that he has the intention to do γ . Then we should have $P(Int(\alpha, s_2)) > P(Int(\alpha, s_1))$ and $P(Int(\gamma, s_2)) > P(Int(\gamma, s_1))$.

It is sensible to assume that $P(Int(\alpha, s_i))$ and $P(Int(\gamma, s_i))$ increase in the same way from s_1 to s_2 .

So, if $\pi(\alpha) = \pi(\beta) = \pi(\gamma)$, $Int(\alpha, s_2)$ and $Int(\gamma, s_2)$ have the same and the greatest probability and the system believes that the agent has the intention to do α and that he has the intention to do γ .

Let us use the following notation.

$BInt(\alpha, s_i)$: in the situation s_i the system believes that the agent has the intention to do α .

Using this notation we have: $BInt(\alpha, s_2)$, $\neg BInt(\beta, s_2)$ and $BInt(\gamma, s_2)$.

We have adopted the following general assumption.

Assumption H3. In a situation s_i such that $Doing(\alpha, s_0, s_i)$, if there is no procedure β such that $Doing(\beta, s_0, s_i)$ and $P(Int(\beta, s_i)) > P(Int(\alpha, s_i))$, then the system believes in s_i that the agent has the intention to do α (i.e. we have $BInt(\alpha, s_i)$).

H3 can be reformulated as:

$BInt(\alpha, s)$ iff $Doing(\alpha, s_0, s)$ and there is no procedure β such that $P(Int(\beta, s)) > P(Int(\alpha, s))$

In the situation $s_3 = do([f, a, d], s_0)$ we have $Doing(\alpha, s_0, s_3)$, $Doing(\beta, s_0, s_3)$ and $Doing(\gamma, s_0, s_3)$.

In s_3 we can assume that $P(Int(\beta, s_i))$ has increased from s_2 to s_3 in the same way as $P(Int(\alpha, s_i))$ and $P(Int(\gamma, s_i))$ have increased from s_1 to s_2 .

For the procedures α and γ , in s_2 the agent has the choice between doing the next recommended action (that are respectively b and f) or doing any other

⁶ To be more precise we should say that the agent has the intention to reach a situation where α has been done.

action. We have assumed that if he does not do the recommended action, then the probability to do the corresponding procedure decreases, because the last observed action does not confirm that he has the intention to do this procedure.

Then, if $\pi(\alpha) = \pi(\beta) = \pi(\gamma)$ we have: $P(Int(\alpha, s_3)) < P(Int(\beta, s_3))$ and $P(Int(\gamma, s_3)) < P(Int(\beta, s_3))$, and therefore we have $BInt(\beta, s_3)$, $\neg BInt(\alpha, s_3)$ and $\neg BInt(\gamma, s_3)$.

In the situation $s_4 = do([f, a, d, b], s_0)$ we have $Doing(\alpha, s_0, s_4)$, $Doing(\beta, s_0, s_4)$ and $Doing(\gamma, s_0, s_4)$.

In that situation the action b is a recommended action for α but it is not a recommended action for γ . Then, if $\pi(\alpha) = \pi(\gamma)$ we should have $P(Int(\alpha, s_4)) > P(Int(\gamma, s_4))$.

If we compare the procedures α and β in s_4 , there are two performed actions (a and b) that are recommended in α , and there is only one (a) which is recommended in β . The number of performed actions that are not recommended is the same for α and β (action d for α and action b for β). Therefore, if $\pi(\alpha) = \pi(\beta)$ we should have $P(Int(\alpha, s_4)) > P(Int(\beta, s_4))$. Then, we have $BInt(\alpha, s_4)$, $\neg BInt(\beta, s_4)$ and $\neg BInt(\gamma, s_4)$.

In the situation $s_5 = do([f, a, d, b, c], s_0)$ we have $\neg Doing(\alpha, s_0, s_5)$ (because α has been executed), $Doing(\beta, s_0, s_5)$ and $Doing(\gamma, s_0, s_5)$.

The number of recommended actions is 1 for β and γ in s_5 , but the number of not recommended actions is 3 for γ and 2 for β . Then, if $\pi(\alpha) = \pi(\beta) = \pi(\gamma)$ we should have $P(Int(\beta, s_5)) > P(Int(\gamma, s_5))$ and $P(Int(\beta, s_5)) > P(Int(\alpha, s_5))$. Therefore we have $BInt(\beta, s_5)$, $\neg BInt(\alpha, s_5)$ and $\neg BInt(\gamma, s_5)$.

From this example we can derive some general guidelines that are expressed with the following terminology.

In a procedure definition we call an action a **prescribed** action if that action explicitly appears in the procedure and it is just preceded by an explicit action.

For example, if α has the form: $\dots; a; b; \dots$ then this occurrence of b is a prescribed action in α . Notice that in a given procedure some occurrences of b may be prescribed actions and others not, like in $\alpha = c; \sigma; b; a; b$.

In a procedure definition we call an action a **recommended** action if that action explicitly appears in the procedure and it is just preceded by a term of the form σ or σ/β .

For example, if α has the form: $\dots; \sigma; a; \dots$ or $\dots; \sigma/(b|c); a; \dots$ then this occurrence of a is a recommended action in α .

Let us call A the set of actions that can be done by the agent and can be observed by the system.

In a procedure definition we call an action a **tolerated** action if the procedure has the form: $\dots; \sigma; a; \dots$ and this action is in $A - \{a\}$.

For example, if $A = \{a, b, c, d, e\}$ and α has the form: $\dots; \sigma; a; \dots$, then the set of tolerated actions for this occurrence of σ is $\{b, c, d, e\}$.

In a procedure definition we call an action a **restricted tolerated** action if the procedure has the form: $\dots; \sigma/(a_{i_1} | \dots | a_{i_i}); a; \dots$ and this action is in $A - \{a_{i_1}, \dots, a_{i_i}, a\}$.

For example, if α has the form: $\dots; \sigma/(b|d); a; \dots$ the set of restricted tolerated actions for this occurrence of σ is $\{c, e\}$.

With these definitions we can formulate our basic guidelines in that way.

Guideline A. If in the situation s_i the last performed action is a prescribed action of α , then $P(Int(\alpha, s_i))$ should be much greater than $P(Int(\alpha, s_{i-1}))$.

Guideline B. If in the situation s_i the last performed action is a recommended action of α , then $P(Int(\alpha, s_i))$ should be greater than $P(Int(\alpha, s_{i-1}))$, but it should be less greater than in the case of a prescribed action.

Guideline C. If in the situation s_i the last performed action is a tolerated action of α , then $P(Int(\alpha, s_i))$ should be lower than $P(Int(\alpha, s_{i-1}))$.

Guideline D. If in the situation s_i the last performed action is a restricted tolerated action of α , then the fact that $P(Int(\alpha, s_i))$ is greater or lower than $P(Int(\alpha, s_{i-1}))$ depends on the cardinality of the set of restricted tolerated actions.

We also have adopted the following assumption about the evolution of the fact that the agent has the intention to do a procedure α .

Assumption H4. In a situation s_i such that we have $Doing(\alpha, s_0, s_i)$ it is assumed that the agent has in s_i the intention to do α iff he has the intention to do α in s_{i-1} .

The assumption H4 is expressed in formal terms as follows.

$$(H4) \quad \forall s \forall s' \forall s'' \forall a \forall \alpha ((Doing(\alpha, s, s'') \wedge s'' = do(a, s')) \rightarrow (Int(\alpha, s'') \leftrightarrow Int(\alpha, s')))$$

H4 is logically equivalent to the conjunction of H'4 and H''4.

$$(H'4) \quad \forall s \forall s' \forall s'' \forall a \forall \alpha (Doing(\alpha, s, s'') \wedge s'' = do(a, s') \wedge Int(\alpha, s') \rightarrow Int(\alpha, s''))$$

$$(H''4) \quad \forall s \forall s' \forall s'' \forall a \forall \alpha (Doing(\alpha, s, s'') \wedge s'' = do(a, s') \wedge Int(\alpha, s'') \rightarrow Int(\alpha, s'))$$

The assumption H'4 means that the agent's intention is persistent as long as the procedure α is not completely performed. That corresponds to the notion of intention persistence proposed by Cohen and Levesque in [9] (see also [16]).

The assumption H''4 corresponds to a different idea. This idea is that if the action a performed by the agent is consistent with the fact that he is doing α and in the situation s'' the agent has the intention to do α , then he has performed the action a **because** in s' he had the intention to do α .

4.2 General method to compute the probabilities

To present the general method we shall use the following notations.

$A = \{a_1, a_2, \dots, a_N\}$: set of actions that can be performed by the agent and that can be observed by the system.

We adopt the following assumption.

Assumption H5. It is assumed that in the language definition the set of atomic action constant symbols is A .

The assumption H5 intuitively means that the actions performed by the agent that cannot be observed by the system are ignored by the system. This assumption is consistent with the fact that what the system believes about the agents' intentions is only founded on his observations.

o_i : i th observation action performed by the system.
 $a_{j_i} = obs(o_i)$: a_{j_i} is the action performed by the agent that has been observed by the system by means of the observation action o_i .

O_i : short hand to denote the proposition $a_{j_i} = obs(o_i)$.

$$O_{1,i} \stackrel{\text{def}}{=} O_1 \wedge O_2 \wedge \dots \wedge O_i$$

$$O_{1,0} \stackrel{\text{def}}{=} true$$

s_0 : initial situation.

$$s_i = do(a_{j_i}, s_{i-1})$$

$P(Int(\alpha, s_i)|O_{1,i})$: probability that in the situation s_i the agent has the intention to do α if the sequence of observations is $O_{1,i}$.

From Bayes' theorem we have:

$$(1) P(Int(\alpha, s_i)|O_{1,i}) = \frac{P(O_{1,i}|Int(\alpha, s_i)) \times P(Int(\alpha, s_i))}{P(O_{1,i})}$$

From (1) we have:

$$(2) P(Int(\alpha, s_i)|O_{1,i}) = \frac{P(O_i \wedge O_{1,i-1}|Int(\alpha, s_i)) \times P(Int(\alpha, s_i))}{P(O_i \wedge O_{1,i-1})}$$

Then, we have:

$$(3) P(Int(\alpha, s_i)|O_{1,i}) = \frac{P(O_i|O_{1,i-1} \wedge Int(\alpha, s_i))}{P(O_i|O_{1,i-1})} \times \frac{P(O_{1,i-1}|Int(\alpha, s_i)) \times P(Int(\alpha, s_i))}{P(O_{1,i-1})}$$

If $\neg Doing(\alpha, s_0, s_i)$:

From H2 we have: $P(Int(\alpha, s_i)|O_{1,i}) = P(Int(\alpha, s_i))$. Then we have:

$$(4) P(Int(\alpha, s_i)|O_{1,i}) = \pi(\alpha)$$

If $Doing(\alpha, s_0, s_i)$:

From H4 we have: $Int(\alpha, s_i) \leftrightarrow Int(\alpha, s_{i-1})$.

Then, from (3) we have:

$$(5) P(Int(\alpha, s_i)|O_{1,i}) = \frac{P(O_i|O_{1,i-1} \wedge Int(\alpha, s_{i-1}))}{P(O_i|O_{1,i-1})} \times \frac{P(O_{1,i-1}|Int(\alpha, s_{i-1})) \times P(Int(\alpha, s_{i-1}))}{P(O_{1,i-1})}$$

Therefore we have:

$$(6) P(Int(\alpha, s_i)|O_{1,i}) = \frac{P(O_i|O_{1,i-1} \wedge Int(\alpha, s_{i-1}))}{P(O_i|O_{1,i-1})} \times P(Int(\alpha, s_{i-1})|O_{1,i-1})$$

If we adopt the notations:

$$num_i(\alpha) \stackrel{\text{def}}{=} P(O_i|O_{1,i-1} \wedge Int(\alpha, s_{i-1}))$$

$$den_i(\alpha) \stackrel{\text{def}}{=} P(O_i|O_{1,i-1})$$

$$F_i(\alpha) \stackrel{\text{def}}{=} \frac{num_i(\alpha)}{den_i(\alpha)}$$

We have:

$$(7) P(Int(\alpha, s_i)|O_{1,i}) = F_i(\alpha) \times P(Int(\alpha, s_{i-1})|O_{1,i-1})$$

The formula (7) allows to regress the computation of $P(Int(\alpha, s_i)|O_{1,i})$ until a situation s_j where we have $\neg Doing(\alpha, s_0, s_j)$ ⁷.

4.3 Heuristics to estimate the probabilities

To define heuristics to estimate the value of $F_i(\alpha)$ we have restricted the set of procedures to procedures of the form:

$$\alpha = A_1; \Sigma_1; \dots; A_k; \Sigma_k; A_{k+1}; \dots; A_s$$

where each A_k denotes an atomic action in A and Σ_k either is absent or denotes a term of the form $\sigma/(a_{i_1} | \dots | a_{i_l})$ where each a_{i_j} is in A and l may be equal to 0. This form will be called in the following: "linear normal form".

⁷ Notice that for any procedure α we have $\neg Doing(\alpha, s_0, s_0)$.

Notice that this form is not a too strong restricted form because a procedure can be transformed by repeatedly applying the transformation rule that transforms $\alpha_1; (\alpha_2|\alpha_3); \alpha_4$ into $(\alpha_1; \alpha_2; \alpha_4)|(\alpha_1; \alpha_3; \alpha_4)$. At the end we get a procedure in the form $\alpha = \alpha_1|\alpha_2|\dots|\alpha_p$. Then, the only difference between each α_i and a procedure in linear normal form is that the A_k s may denote either an atomic action or a test action, and the Σ_k s, when they are not absent, have in general the form σ/β where β may be any kind of procedure.

Now we are going to define the estimation of the term $F_i(\alpha)$ in the case where we have $Doing(\alpha, s_0, s_i)$.

The estimation of $F_i(\alpha)$ depends on the part α'_{i-1} of α which has already been performed in the situation s_{i-1} . This part is defined by the property $Done(\alpha'_{i-1}, \alpha, s_0, s_i)$ where the property $Done$ is defined as follows.

$$Done(\alpha', \alpha, s, s') \stackrel{\text{def}}{=} Doing(\alpha, s, s') \wedge start(\alpha', \alpha) \wedge \exists s_1 (s \leq s_1 < s' \wedge Do_s(\alpha, s_1, s') \wedge Do_p(\alpha', s_1, s'))$$

In this definition the condition $Do_s(\alpha, s_1, s')$ guarantees that the part of α that is being performed in s has started his performance in s_1 , and the condition $Do_p(\alpha', s_1, s')$ guarantees that there is no part of α that is longer than α' that has been performed between s_1 and s' . $Done(\alpha', \alpha, s, s')$ intuitively means that α' is the maximal part of α that has started between s and s' and that has ended in s' .

For instance, in the previous example in s_2 we have $Doing(\alpha, s_0, s_2)$ and for $\alpha'_2 = a$ we have $Do_s(\alpha'_2, s_1, s_2)$ and $Do_p(\alpha'_2, s_1, s_2)$. In s_3 we have $\alpha'_3 = a; \sigma$ and in s_4 we have $\alpha'_4 = a; \sigma; b$.

To estimate $F_i(\alpha)$ we have accepted the following assumption.

Assumption H6. It is assumed that the i th observation O_i is independent of the previous observations and each action in A has the same probability to be observed.

In formal terms H6 is expressed by: $den_i(\alpha) = P(O_i|O_{1,i-1}) = P(O_i) = \frac{1}{N}$.

We shall use the notation $O_i = A_k$ to express that the action a_{i_j} observed by the observation action o_i is the atomic action denoted by A_k , and we use the notation $O_i \in \Sigma_k$ to express that a_{i_j} is in the set $A - \{a_{i_1}, \dots, a_{i_i}, a_{k+1}\}$, where a_{k+1} is the action denoted by A_{k+1} .

The terms $num_i(\alpha)$ and $F_i(\alpha)$ have to be estimated only in the case where we have $Doing(\alpha, s_0, s_i)$. We have to consider different cases.

Case 1. We have $\neg Doing(\alpha, s_0, s_{i-1})$.

In that case $\alpha'_{i-1} = A_1$ and, from the assumption H1, $Int(\alpha, s_{i-1})$ and $Doing(\alpha, s_0, s_i) \wedge \neg Doing(\alpha, s_0, s_{i-1})$ imply that in s_i the agent has performed the action A_1 , and the observed action in O_i is A_1 . Then, we necessarily have $O_i = A_1$.

Therefore we have $num_i(\alpha) = 1$ and $F_i(\alpha) = N$.

Case 2. We have $Doing(\alpha, s_0, s_{i-1})$.

– **Case 2.1.** α'_{i-1} has the form $\alpha'_{i-1} = \dots; A_k$.

- **Case 2.1.1.** α has the form
 $\alpha = \dots; A_k; A_{k+1}; \dots$

In that case Σ_k is absent in α . From the assumption H1, $Int(\alpha, s_{i-1})$ implies that the action performed in s_i is A_{k+1} . Then, we necessarily have $O_i = A_{k+1}$.

Therefore we have $num_i(\alpha) = 1$ and $F_i(\alpha) = N$.

- **Case 2.1.2.** α has the form

$\alpha = \dots; A_k; \Sigma_k; A_{k+1}; \dots$

Case 2.1.2.1. $O_i = A_{k+1}$.

The general form of Σ_k is $\sigma/(a_{i_1} | \dots | a_{i_l})$.

According to guideline B it is much more likely that the action performed by the agent in s_i is the recommended action A_{k+1} than any restricted tolerated action defined by Σ_k .

Then we have $num_i(\alpha) = 1 - \epsilon$ where the value of ϵ is defined in function of the application domain and is supposed to be “small” with respect to 1.

We have $F_i(\alpha) = N \times (1 - \epsilon)$.

Case 2.1.2.2. $O_i \neq A_{k+1}$.

Here we have adopted the following assumption.

Assumption H7. It is assumed that when the agent has the intention to do α all the restricted tolerated actions have the same probability to be performed by the agent.

According to H7 any action in $A - \{a_{i_1}, \dots, a_{i_l}, a_{k+1}\}$ has the same probability to be done. Then, we have⁸: $num_i(\alpha) = \frac{\epsilon}{N-(l+1)}$.

We have $F_i(\alpha) = \frac{N}{N-(l+1)} \times \epsilon$.

- **Case 2.2.** α'_{i-1} has the form $\alpha'_{i-1} = \dots; \Sigma_k$.

Case 2.2.1. $O_i = A_{k+1}$.

We are in the same type of situation as in the case 2.1.2.1. Then we have $num_i(\alpha) = 1 - \epsilon$ and $F_i(\alpha) = N \times (1 - \epsilon)$.

Case 2.2.2. $O_i \neq A_{k+1}$.

We are in the same type of situation as in the case 2.1.2.2. Then we have $num_i(\alpha) = \frac{\epsilon}{N-(l+1)}$ and $F_i(\alpha) = \frac{N}{N-(l+1)} \times \epsilon$.

In the case where the action that has been performed by the agent in s_i is a prescribed action (cases 1. and 2.1.1.) we have $F_i(\alpha) = N$. This conforms the guideline A.

In the case where the performed action is a recommended action (cases 2.1.2.1. and 2.2.1.) we have $F_i(\alpha) = N \times (1 - \epsilon)$. To fulfill the guideline B, that is: $F_i(\alpha) > 1$, we have to assign to ϵ a value such that $\epsilon < \frac{N-1}{N}$.

In the case where the performed action is a tolerated action (cases 2.1.2.2. and 2.2.2. and $l = 0$) we have $F_i(\alpha) = \frac{N}{N-1} \times \epsilon$. From the assumption $\epsilon < \frac{N-1}{N}$ we have $F_i(\alpha) < 1$ and this fulfills the guideline C.

In the case where the performed action is a restricted tolerated action (cases 2.1.2.2. and 2.2.2. and $l > 0$) we have $F_i(\alpha) = \frac{N}{N-(l+1)} \times \epsilon$.

⁸ Notice that the case $N - (l + 1) = 0$ can be rejected because if $l = N - 1$ there is only one restricted tolerated action and the agent has no choice offered by Σ_k .

Therefore we have $F_i(\alpha) < 1$ iff $\epsilon < \frac{N-(l+1)}{N}$ (therefore we also have $\epsilon < \frac{N-1}{N}$), and we have $F_i(\alpha) > 1$ iff $\epsilon > \frac{N-(l+1)}{N}$ (and this is consistent with $\epsilon < \frac{N-1}{N}$).

Therefore, according to guideline D we may have either $F_i(\alpha) < 1$ or $F_i(\alpha) > 1$ depending on the values of ϵ , l and N .

4.4 Coming back to the example

The method we have presented can be used to compute **iteratively** the values of $P(Int(\alpha, s_i)|O_{1,i})$, $P(Int(\beta, s_i)|O_{1,i})$ and $P(Int(\gamma, s_i)|O_{1,i})$.

If we use the notations $\Pi_i(\alpha) = P(Int(\alpha, s_i)|O_{1,i})$, $\Pi_i(\beta) = P(Int(\beta, s_i)|O_{1,i})$, $\Pi_i(\gamma) = P(Int(\gamma, s_i)|O_{1,i})$, and $R = N \times (1 - \epsilon)$, for recommended actions, and $T = \frac{N}{N-1} \times \epsilon$ for tolerated actions we get the following table

s_i	$\Pi_i(\alpha)$	$\Pi_i(\beta)$	$\Pi_i(\gamma)$
s_0	$\pi(\alpha)$	$\pi(\beta)$	$\pi(\gamma)$
s_1	$\pi(\alpha)$	$\pi(\beta)$	$\pi(\gamma)$
s_2	$N \times \pi(\alpha)$	$\pi(\beta)$	$N \times \pi(\gamma)$
s_3	$N \times T \times \pi(\alpha)$	$N \times \pi(\beta)$	$N \times T \times \pi(\gamma)$
s_4	$N \times R \times T \times \pi(\alpha)$	$N \times T \times \pi(\beta)$	$N \times T^2 \times \pi(\gamma)$
s_5	$\pi(\alpha)$	$N \times T^2 \times \pi(\beta)$	$N \times T^3 \times \pi(\gamma)$

We have $N > R > 1$ and $T < 1$.

If we have $\pi(\alpha) = \pi(\beta) = \pi(\gamma)$ we can determine what the system believes about the agents' intentions in these situations. As expected in 4.1 we get:

In s_0 we have $BInt(\alpha, s_0)$, $BInt(\beta, s_0)$ and $BInt(\gamma, s_0)$.

In s_1 we have $BInt(\alpha, s_1)$, $BInt(\beta, s_1)$ and $BInt(\gamma, s_1)$.

In s_2 we have $BInt(\alpha, s_2)$ and $BInt(\gamma, s_2)$.

In s_3 we have $BInt(\beta, s_3)$.

In s_4 we have $BInt(\alpha, s_4)$.

In s_5 we have $BInt(\beta, s_5)$.

5 Conclusion

We have presented a method to assign intentions to an agent which is based on the computation of the estimation of the probability that an agent has the intention to perform a procedure.

There are two parts in the computation method. The first part (section 4.2) is general and is based on the assumptions H1-H4. The second part (section 4.3) is based on heuristics and on the additional assumptions H5-H7 and requires to know the value of $\pi(\alpha)$ for each α . The values of N and l are determined by the application domain and the value of ϵ can be tuned by a designer.

A difference with other methods for plan recognition is that in the procedures we may have terms of the form σ/β . The property *Doing* allows the selection of the procedure that matches the observations $O_{1,i}$. To estimate the probability of the occurrence of the next observation O_i we consider the part α'_{i-1} of the

procedure α that has already been performed. Therefore the estimated probabilities depend on the history and not just on the previous observation O_{i-1} . This is an important original aspect of the method.

The computation cost of the estimated probabilities and of the evaluation of the properties *Doing* and *Done* is linear with respect to the number of observations for a given procedure. That makes the computation very fast.

Finally, it is worth noting that a preliminary version of the method has been implemented in Prolog [11]. This implementation was of great help to check our intuition on simple examples.

Future works will be:

- 1) to remove the too strong assumption H6 about the independence of the observations O_i in order to have a better estimation of $\frac{P(O_i|O_{1,i-1} \wedge Int(\alpha, s_{i-1}))}{P(O_i|O_{1,i-1})}$,
- 2) to guarantee that after a long sequence of observations of tolerated actions $P(Int(\alpha, s_i)|O_{1,i})$ is never lower than $\pi(\alpha)$ and
- 3) to allow test actions $\phi?$ and temporal conditions in the procedure definitions.

Acknowledgment. We are very grateful to G. Eizenberg for his help in Probability Theory. If there are some errors they are the own responsibility of the authors.

References

1. D. W. Albrecht, I. Zukerman, and A. E. Nicholson. Bayesian models for key-hole plan recognition in an adventure game. In *User modeling and user-adapted interaction*, volume 8, pages 5–47. 1998.
2. D. E. Appelt and M. E. Pollack. Weighted abduction for plan ascription. In *User modeling and user-adapted interaction*, volume 2, pages 1–25. 1991.
3. J. A. Baier. On procedure recognition in the Situation Calculus. In *22nd International Conference of the Chilean Computer Science Society*. IEEE Computer Society, 2002.
4. M. Bauer. Integrating probabilistic reasoning into plan recognition. In *Proceedings of the 11th European Conference of Artificial Intelligence*. John Wiley and Sons, 1994.
5. N. Blaylock and J. Allen. Corpus-based, statistical goal recognition. In G. Gottlob and T. Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 1303–1308, 2003.
6. C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic high-level agent programming in the situation calculus. In *Proceedings of AAAI*. 2000.
7. S. Carberry. Incorporating default inferences into plan recognition. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 471–478. 1990.
8. E. Charniak and R. P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1), 1993.
9. P. R. Cohen and H. J. Levesque. Persistence, Intention, and Commitment. In A. L. Lansky M. P. Georgeff, editor, *Reasoning about actions and plans*, pages 297–340, Timberline, USA, 1986.
10. R. Demolombe and E. Hamon. What does it mean that an agent is performing a typical procedure? A formal definition in the Situation Calculus. In C. Castelfranci and W. Lewis Johnson, editor, *First International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM Press, 2002.

11. A. M. Otermin Fernández. Reconocimiento de intenciones de un operador que interacta con un sistema. Technical Report, ONERA Toulouse, 2004.
12. A. Finzi and T. Lukasiewicz. Game theoretic GOLOG under partial observability. In F. Dignum, V. Dignum, S. Koenig, S. Kraus, and M. Wooldridge, editors, *Proceedings of the 4th International Conference on Autonomous Agents and Multi Agent Systems*. ACM Press, 2005.
13. H. A. Kautz. A formal theory of plan recognition and its implementation. In J. F. Allen, H. A. Kautz, R. N. Pelavin, and J. D. Tennenberg, editors, *Reasoning about plans*, pages 69–126. Morgan Kaufman Publishers, 1991.
14. H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming*, 31:59–84, 1997.
15. W. Mao and J. Gratch. A utility-based approach to intention recognition. In *Proceedings of the AAMAS 2004 Workshop on Agent Tracking: modeling other agents from observations*, 2004.
16. P. Pozos Parra, A. Nayak, and R. Demolombe. Theories of intentions in the framework of Situation Calculus. In *Proceedings of the AAMAS workshop on Declarative Agent Languages and technologies*, 2004.
17. R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.