Intention Recognition with Event Calculus Graphs

Fariba Sadri
Imperial College, London, UK
fs@doc.ic.ac.uk

Abstract

Intention recognition has significant applications in ambient intelligence, for example in assisted living and care of the elderly, in games and in intrusion and other crime detection. In this paper we propose an intention recognition system based on the event calculus. The system, called WIREC, exploits profiles, contextual information, heuristics and any available integrity constraints together with plan libraries and a basic theory of actions, causality and ramifications. Whenever the profile and context suggest there is a usual pattern of behaviour on the part of the actor the search for intention can be focused on existing plan libraries. On the other hand, when no such information is available or if the behaviour of the actor deviates from the usual pattern the search for intention can revert to the basic theory of actions, in effect dynamically constructing partial plans corresponding to the actions executed by the actor.

1. Introduction

Intention recognition is the task of recognizing the intentions of an agent by analyzing their actions and/or analyzing the changes in the state (environment) resulting from their actions. Research on intention recognition has been going on for the last 30 years or so. Early applications included story understanding and automatic response generation, for example in Unix help facilities. Examples of early work can be found in [11, 20]. More recently a host of new applications of intention recognition has attracted much interest.

These more recent applications include assisted living and ambient intelligence, increasingly sophisticated computer games, and intrusion and terrorism detection and more militaristic applications. All these have brought new and exciting challenges to the field. For example assisted living applications require recognizing the intentions of residents in domestic environments in order to anticipate and assist with their needs. Applications in computer systems intrusion or terrorism detection require recognizing the intentions of the would-be-attackers in order to prevent them. Military applications need recognizing the intentions of the enemy maneuvers in order to plan counter-measures and react appropriately.

Examples of literature on intention recognition for these applications are [16] and [7], for the care of the elderly, [8] and [18] for assistance for cognitively impaired individuals, for example Alzheimer patients, [6] for computer system intrusion detection, [10] for terrorism detection, [3] for real-time computer strategy games, [15] for anticipating military movements, and [21] for riot control in urban environments.

Cohen, et al. [4] classify intention recognition as either *intended* or *keyhole*. In the *intended* case the agent which is being observed wants his intentions to be identified and intentionally gives signals to be sensed by other (observing) agents. In the *keyhole* case the agent which is being observed either does not intend for his intentions to be identified, or does not care; he is focused on his own activities, which may provide only partial observability to other agents. Our approach is applicable to both cases of intention recognition, but we describe it for the first case only.

The intention recognition problem has been cast in different formalisms and methodologies. Prominent amongst these are logic-based, case-based and probabilistic approaches. Regardless of the formalism, much of the work on intention recognition is based on using pre-specified plan libraries that aim to predict the intentions and plans of the actor agent. Use of the plan libraries has obvious advantages, amongst them managing the space of possible hypotheses about the actor's intentions. But it also has a number of limitations. For example anticipating, acquiring and coding the plan library are not easy tasks, and if intention recognition relies entirely on plan libraries then it cannot deal with cases where the actor's habits are not well-known or if the actor exhibits new, unanticipated behaviour.

In this paper we propose a new logic-based approach to intention recognition based on deduction and the Event Calculus (EC) [13] which is a formalism for reasoning about events, causality and ramifications. The contributions of the paper are as follows. It proposes a system called WIREC (Weighted Intention Recognition based on Event Calculus). WIREC exploits any available information about the actor, his actions, the context, including the actor's context-based usual behaviour, and constraints, for example his inability to perform certain tasks in certain circumstances.

WIREC can exploit plan libraries if any plans correspond to the known profile of the actor, and it can revert to a basic theory of causality if no such plans are available or if the actor's behaviour deviates from his known profile. We also briefly describe how it incorporates a concept of "weight-of-evidence" to focus the search for intentions and to rank the hypotheses about intentions. A longer paper [19] gives details on this, and also describes how WIREC takes into account the actor's knowledge-seeking actions, as well as his physical actions, and reasons with what it infers about the actor's knowledge.

Chen et al. [2] also use the event calculus for reasoning about intentions and actions, in a framework for assisted living, but in their work they know the intention of the actor a priori, and use the event calculus to plan for the intention in order to guide the actor through the required actions. Hong [9] shares with us concerns about the limitations of intention recognition being based entirely on plan libraries. In his work he does not use plan libraries and uses a form of graph search through state changes. But his aim is to identify fully or partially achieved goals, by way of explaining executed actions rather than to predict future intentions and actions.

2. Motivating Example

Suppose John is at home and we observe that he is boiling some water. The immediate intention is, of course, to have boiled water. But several "longer term" intentions may be possible, for example to make a meal or to make a hot drink. Several factors can help us narrow the space of possible hypotheses about John's intentions and to rank them. We focus on two factors.

One factor is any knowledge about the current context and about John's profile. For example if it is 9 am and John "normally" has tea around this time, then one reasonable possible intention to investigate is that of John having tea. On the other hand, if it is a hot day and John does not have hot drinks when it is hot, then it would be reasonable to consider other intentions instead.

The other factor is "weight of evidence", which can be used if John's profile is not known, or in conjunction with his profile, or if John is behaving in a way unanticipated by his known profile. Weight of evidence is based on what we observe John do. John may perform "knowledge seeking" actions. Observing such actions provides us with information about what he knows, and that can be used in calculating weight of evidence. For example if before boiling the water John opens the cupboard where the tea and other groceries are kept and looks inside, then we know that he knows the status of the tea. In particular if we also know that there is no tea in the cupboard (via RFID tag readers, say) then that eliminates the possibility that John wants to make tea.

John may also perform further "physical" actions. For example if after boiling the water he opens the cupboard where the pasta is kept then that lends weight to the hypothesis that he intends to make a meal. On the other hand if after boiling the water he takes the water to the sink then that weakens the meal hypothesis and strengthens the possibility that he wants to pour the water down the sink, possibly to unblock the drain.

3. Background

The approach we take in this paper is based on the event calculus (EC). This formalism allows us to specify the semantics of actions in terms of their preconditions and the fluents (time-dependent properties) they initiate and terminate. EC has been used for planning, by [14], for example. The ontology contains a set of action operators, symbolized by *A*, *a*, *a1*, *a2*, *b*, *c*, etc, a set of fluents, symbolized by *P*, *p*, *p1*, *p2*, .., *q*, *r*, *neg(p)*, etc, and a set of time points. Initiation, termination and preconditions can be specified by domain-dependent rules of the form:

Initiation:
$initiates(A,P,T) \leftarrow holds(P_1,T) \wedge \ldots \wedge holds(P_n,T)$
Termination:
$terminates(A,P,T) \leftarrow holds(P_1,T) \wedge \ldots \wedge holds(P_n,T)$
Precondition: $precondition(A,P)$

The first two rules, above, state that action *A* initiates (resp. terminates) fluent *P* at time *T* if fluents P_1, ..., P_n hold at *T*. The conditions $holds(P_1,T) \wedge \ldots \wedge holds(P_n,T)$, above, are called *qualifying conditions*. The fluents *P* in the conclusion of initiation and termination rules are called *primitive* fluents.

Further we can specify how actions affect primitive fluents (using the *holds* predicate). We give some of the rules below. In these rules all variables are assumed universally quantified in front of the rule, unless specified otherwise. The first rule states that a fluent P holds at time T_2 if an action A initiating it is done at an earlier time T_1, and all of the action's preconditions held at that time, and the fluent P has not been *clipped* in the interval between T_1 and T_2. A fluent is clipped in a time interval if an action occurs in that interval that terminates the fluent.

$$holds(P,T_2) \leftarrow do(A,T_1) \wedge initiates(A, T_1, P) \wedge T_1<T_2 \wedge (\forall P \quad precondition(A,P) \rightarrow holds(P, T_1)) \wedge not\ clipped(T_1, P, T_2)$$

$$clipped(T_1, P, T_2) \leftarrow do(B,T) \wedge terminates(B, T, P) \wedge T_1 < T \wedge T =< T_2$$

Finally we can specify ramifications, i.e. fluents holding as a result of others that hold. To do so we use domain-dependent rules of the form:

Ramification:
$holds(Q, T) \leftarrow holds(P_1,T) \wedge \ldots \wedge holds(P_n,T)$

As an example of EC specification consider the following (self-explanatory) domain-dependent rules:

Example 1.
initiates(pushOnButton(Actor, radio), on(radio), T) ← holds(hasBattery(radio), T) ∧ holds(neg(on(radio)), T)
terminates(pushOnButton(Actor, radio), on(radio), T) ← holds(on(radio),T)
precondition(pushOnButton(Actor,radio), co-located(Actor, radio))
holds(co-located(X,Y), T) ← holds(loc(X,L), T) ∧ holds(loc(Y,L),T)

4. Intention recognition: our approach

We make the following assumptions. There are two agents, the observer (which is the WIREC system), and the actor, who is assumed to be rational, and may have multiple (concurrent) intentions. We observe the actions of the actor in the order they take place, and the actions are successfully executed.

Although our approach works with both full and partial observability, here, for simplicity, we deal with the former only. As well as actions, we also observe fluents. In an ambient intelligence assisted living scenario, for example, the house will have a collection of sensors, and readings from these can periodically update the representation of state kept by the system. In our work such observed fluents will typically be properties that can change without the intervention of the actor, for example, whether the actor is alone or has company, whether it is a hot day, and so on. Note that observation of fluents also facilitates dealing with partial observability of actions, not explored here.

In this paper an intention may be an action or a fluent. In the former case, the actor's actions are directed towards achieving the preconditions of the intended action, thus making the action executable. In the latter case the actor's actions are directed towards achieving the intended fluent.

4.1. Graph representation of the event calculus

In this work we adopt a graph-like representation of the event calculus axioms (and plans). This is given in Table 1. Each instance of a graph given in the last column is called a *graph fragment*. This graphic representation allows our intention recognition algorithm to be interpreted both in terms of reasoning and in terms of graph matching and path finding.

Table 1. EC graph-like representation

EC Axiom Name	EC Axiom schema	Graph Representation
Initiation	initiates(A,P,T) ← holds(P_1,T) ∧ … ∧ holds(P_n,T)	A, P_1, ., ., P_n → P
Termination	terminates(A,P,T) ← holds(P_1,T) ∧ … ∧ holds(P_n,T)	A, P_1, ., ., P_n → P
Precondition	precondition(A,P_1) precondition(A,P_2) . . precondition(A,P_n) being all the precondition axioms for A	P_1, ., ., P_n → A
Ramification	holds(Q, T) ← holds(P_1,T) ∧ … ∧ holds(P_n,T)	P_1, ., ., P_n → Q

Plans (and thus plan libraries) can be constructed using this graph-like representation. For example Fig.

1(i) shows a plan for achieving *r* by doing actions *a1*, *a2*, *a3* in any order, and doing *a4* after *a1* and *a2*. Fig. 1(ii) gives a more conventional representation of the same plan used by other intention recognition systems. The approach in Fig. 1(i) compared to Fig. 1(ii) and to other approaches such as the Hierarchical Task Network models [5] has a number of advantages.

The representation in Fig. 1(i) provides information about qualifying conditions (*p1* and *p2* for the initiation of *q1*), preconditions (*q1* and *q2* for the executability of action *a4*) and ramifications (*r* holding as a result of *r1* and *r2*). All this information can be useful in intention recognition. For example if the observer knows that the actor knows that *p1* does not hold, then if the actor performs action *a1* he certainly does not intend *q1*, nor *a4*, and thus is very unlikely to intend *r*.

Also the observer may not see actions *a1* and *a2* executed, but sees *a4*. The plan makes it clear that *a1* and *a2* are needed only to establish the preconditions for the executability of *a4*. So not having observed them does not distract from the possibility of *r* being an intention. The preconditions of *a4* may have already held and the actor opportunistically executed *a4*.

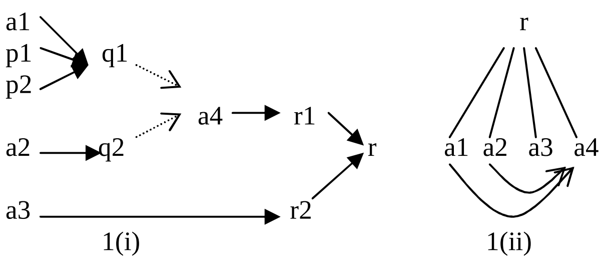

Figure 1(i). An EC plan for achieving intention r
1(ii). A conventional representation of the plan

4.2. Architecture of WIREC

Fig. 2 illustrates the architecture of WIREC. When an action is observed WIREC uses it together with any available Profile and Integrity constraints to update the hypotheses about the intention(s) of the actor. Below we sketch some of the components of WIREC. More detailed descriptions can be found in [19].

4.3. Hypotheses

The set of hypotheses is a set of *weighted entities*, each of the form <*Entity*, *Weight*>, where *Entity* is a ground fluent or ground action operator, and *Weight* is a number between 0 and 1. Each hypothesis represents a possible intention. Intuitively, if <*E*,*W*> is a hypothesis then the actor's actions have contributed towards the (current or future) achievement of *E* (or achievement of the preconditions of *E*, if *E* is an action). Also *W* is a measure of the proportion of the conditions that have already been achieved, typically by the actor's actions, towards the achievement of *E*. Example 2, later, illustrates these intuitions.

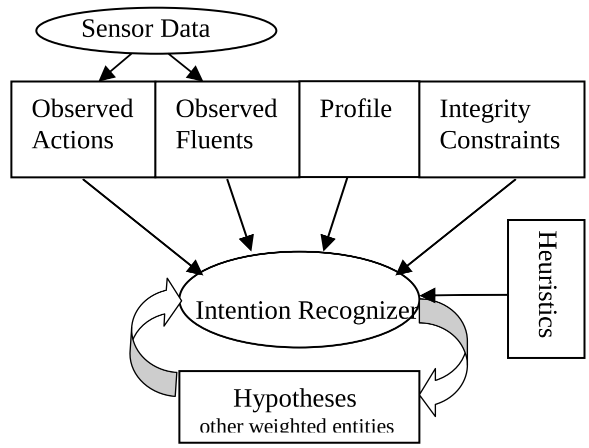

Figure 2. Architecture of WIREC

4.4. Profile and Integrity Constraints

The Profile includes any information available (or acquired through learning) about the actor's usual behaviour in given contexts, in terms of what his intentions may be and how he may go about achieving them. Profile information (and Integrity Constraints) can be specified using (an extension of) the event calculus. For example: If it is cold at night it is possible that John has a hot drink and it is possible that he makes himself a hot-water-bottle:

$$holds(cold, T) \wedge T>22{:}00 \wedge T<1{:}00 \rightarrow pos(have\text{-}hotDrink, T) \wedge pos(have\text{-}hotWaterBottle, T).$$

If we also know how he usually goes about making his drink, for example, we can include the information:

$$pos(have\text{-}hotDrink,T) \wedge T>22{:}00 \wedge T<1{:}00 \rightarrow pos\_plan(p_{herbTea},T) \wedge pos\_plan(p_{cocoa},T),$$

where $p_{herbTea}$ and p_{cocoa} are IDs of plans in the plan library which is part of the Intention Recognizer. Profile can be empty, if nothing is known about the actor.

We assume that the information in the Profile is "positive". "Negative" information, for example about what the actor cannot or would not do, is kept in Integrity Constraints. For example he cannot climb a stool: *do(climb(stool), T) => false*.

4.5. Heuristics

Our heuristics are in two parts: the domain-dependent part and the domain-independent part. The domain-dependent part allows us to distinguish between *consequences* of actions and *intentions* motivating them. An action can have several effects, some of which may be incidental and merely side-effects of the action as far as the actor is concerned. These we call consequences. Other effects may be the (immediate) intentions behind the execution of the action and possibly paving stones towards further actions and longer term intentions.

For example when an actor turns up the thermostat on the water heater, one consequence is that his heating bill goes up, but an immediate intention is that the water temperature increases, and longer term intentions may be to have a bath and get dressed. The distinction between consequences and intentions has been discussed in the literature on double effect and moral computing [12, 17].

The domain-independent part of the Heuristics specifies cut-off points (currently based on a numerical Threshold), beyond which the Intention Recognizer does not look further into possible future intentions.

5. Intention Recognizer

The Intention Recognizer contains several knowledge bases, including S, a representation of the current state of the environment, PL, a (possibly-empty) library of plans, where each plan is of the form of Fig. 1(i), and BL, a library of basic causality theory, consisting of instances of graphs given in Table 1.

When a fluent is observed the Intention Recognizer updates S by assimilating the fluent. When an action is observed the Intention Recognizer first updates S according to the *initiates* and *terminates* axioms of the event calculus, and then proceeds to update the hypotheses about the intentions of the actor. It does so in the following way. First it consults Profile to see if, in the current context (state S), there is any information about the actor's possible intentions and plans, providing an (initial) focus for the search. If so then appropriate plans are selected from PL. If not, or if the sequence of actions observed thus far does not correspond to any plans that may be selected from PL, then the search uses BL. Either way, the search focuses on the executed actions, effectively reasoning forwards from them (which can also be thought of as propagating them through graph matching) and propagating the "weight of evidence". In this process, we also make use of any available Integrity Constraints and heuristic information to prune the search.

Note that when the search uses BL, it amounts to dynamically constructing new partial plans matching the executed actions. Details of the algorithm are given in [19]. Here we illustrate it with an example.

Example 2.

Table 2. Part of BL

2i	2ii	2iii	2iv	2v
p ⇢ a	a → q	q ⇢ b	c → p1	a → t
2vi	**2vii**	**2viii**	**2ix**	**2x**
b → q1 p1 ↗	b → q2 p2 ↗	q2 → r q3 ↗	d → r1 q4 ↗	e → p3 q1 ↗

Suppose BL consists of the fragments in Table 2, where *a,b,c,d,e* are actions, and *p, p1, p2, p3, q, q1, .., q4, r, r1, t* are fluents. Suppose Heuristics informs us that t is a *consequence* and the other fluents can be considered as *intentions*. Fragment 2i and 2iii represent action preconditions, 2viii represents a ramification and the others represent fluent initiations.

Suppose we observe that action *a* has been executed. Reasoning forward from *a* amounts to traversing (some of) the paths starting at *a*. We assign weights as we do the traversal: *<q,1>* (because of 2ii, *q* actually holds now because of *a*), *<b,1>* (2iii, action *b* is enabled - i.e. its precondition(s) now hold - because of *a*), *<q1,1/2>* (2vi, action *b* is enabled by the actor but he has made no effort towards *p1* yet, so only one half of the conditions for achieving *q1* are in place), *<q2,1/2>* (2vii, similar to 2vi), *<r,1/4>* (2viii, the actor has made some effort towards *q2* but none towards *q3* yet), *<p3, 1/4>* (2x, similar to 2viii).

Notice that we ignore 2i, 2iv, 2ix; this is because we focus on the changes that are brought about by the actor. We also ignore 2v because we are interested in changes only if they work towards possible intentions. Furthermore, if we knew that, say, action *e* is not possible for the actor (according to the information in Integrity Constraints) then we would also ignore 2x and not compute a weight for *p3*. Also the weights of *r* (1/4) may be too low according to our Heuristics and we may ignore *r*, and not reason any further with it, for the time being. Now suppose the actor does *c* next. This increases the weight of *q1* to 1 (and *p3* to 1/2 if 2x is still being considered). The other weights remain the same.

Our approach has a flavour of GraphPlan [1], but with two significant differences. Firstly in GraphPlan in each state all actions whose preconditions are satisfied are considered. In our approach we consider only those actions whose preconditions are (fully or partially) satisfied because of the actor's actions. Secondly GraphPlan completely constructs all states as it computes paths into possible futures. We simply partially skim paths into the future. These two considerations, together with the fact that GraphPlan is a fast planner, suggest our approach may have reasonable performance - this is being currently tested.

6. Conclusion and further work

In this paper we proposed an approach to intention recognition based on the event calculus. Ongoing work includes implementation and empirical studies, as well as an investigation into scalability and formal analysis.

References

[1] Blum, A.L., Furst, M.L., "Fast planning through planning graph analysis", *Artificial Intelligence*, 90, 1997, 281-300.

[2] Chen, L., Nugent, C., Mulvenna, M., Finlay, D., Hong, X., Poland, M. A logical framework for behabviour reasoning and assistance in a smart home, *International Journal of Assistive Robotics and Mechatronics*, Vol. 9 No.4, 20-34, 2008.

[3] Cheng, D. C., Thawonmas, R., Case-based plan recognition for real-time strategy games. In Proceedings of the 5th Game-On International Conference (CGAIDE) 2004, Reading, UK, Nov. 2004, 36-40.

[4] Cohen, P.R., Perrault, C.R., Allen, J.F., Beyond question answering. In Strategies for Natural Language Processing, W. Lehnert and M. Ringle (Eds.), Lawrence Erlbaum Associates, Hillsdale, NJ, 1981, 245-274.

[5] Erol, K., Hendler, J., Nau, D. (1994). Semantics for hierarchical task-network planning. CS-TR-3239, University of Maryland, 1994.

[6] Geib, C. W., Goldman, R. P., Plan recognition in intrusion detection systems. In the Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX), June, 2001.

[7] Geib, C. W., Goldman, R. P., Partial observability and probabilistic plan/goal recognition. In Proceedings of the 2005 International Workshop on Modeling Others from Observations (MOO-2005), July 2005.

[8] Giroux S., Bauchet J., Pigot, H., Lusser-Desrochers, D., Lachappelle, Y., Pervasive behavior tracking for cognitive assistance. The 3rd International Conference on Pervasive Technologies Related to Assistive Environments, Petra'08, Greece, July 15-19, 2008.

[9] Hong, Jun, Goal recognition through goal graph analysis, *Journal of Artificial Intelligence Research* 15, 2001, 1-30.

[10] Jarvis, P., Lunt, T., Myers, K., Identifying terrorist activity with AI plan recognition technology. In the Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI 04), AAAI Press, 2004.

[11] Kautz, H., Allen, J.F., Generalized plan recognition. In Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86), 1986, 32-38.

[12] Kowalski, R.A., *How to be artificially intelligent*, Cambridge University Press, to be published.

[13] Kowalski, R.A., Sergot, M.J., "A logic-based calculus of events" In *New Generation Computing*, Vol. 4, No.1, February 1986, 67-95.

[14] Mancarella, P., Sadri, F., Terreni, G., Toni, F., Planning partially for situated agents, 5th Workshop on Computational Logic in Multi-Agent Systems (CLIMA V), 29-30 September 2004, J.Leite and P.Torroni, eds, 132-149.

[15] Mao, W., Gratch, J., A utility-based approach to intention recognition. AAMAS 2004 Workshop on Agent Tracking: Modelling Other Agents from Observations.

[16] Pereira, L.M., Anh, H.T., Elder care via intention recognition and evolution prospection, in: S. Abreu, D. Seipel (eds.), Procs. 18th International Conference on Applications of Declarative Programming and Knowledge Management (INAP'09), Évora, Portugal, November 2009.

[17] Pereira, L. M., Saptawijaya, A., Modelling Morality with Prospective Logic, in: International Journal of Reasoning-based Intelligent Systems (IJRIS), 1(3/4): 209-221, 2009. Also to appear in: M. Anderson, S. Anderson (eds.), "Machine Ethics", Cambridge University Press.

[18] Roy, P., Bouchard B., Bouzouane A, Giroux S., A hybrid plan recognition model for Alzheimer's patients: interleaved-erroneous dilemma. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2007, 131-137.

[19] Sadri, F., Intention recognition with event calculus and weight of evidence, Report, April 2010.

[20] Schmidt, C., Sridharan, N., Goodson, J., The plan recognition problem: an intersection of psychology and artificial intelligence. *Artificial Intelligence*, Vol. 11, 1978, 45-83.

[21] Suzić, R., Svenson, P., Capabilities-based plan recognition. In Proceedings of the 9th International Conference on Information Fusion, Italy, July 2006.