Abductive Inference of Plans and Intentions in Information-Seeking Dialogues

Paulo Quaresma and José Gabriel Lopes

Artificial Intelligence Center, UNINOVA
2825 Monte da Caparica, Portugal
{pq,gpl}@fct.unl.pt

Abstract

A robust man-machine interaction requires the capability for inferring the beliefs and intentions of each active agent. In this article it will be proposed a framework that supports the recognition of plans and intentions through abductive inferences over discourse sentences. The possible actions, world knowledge, events and states are represented by extended logic programs (LP with explicit negation) and the abductive inference process is modeled by the framework proposed by Pereira ([10]) which is based on the Well Founded Semantics augmented with explicit negation (WFSX) and contradiction removal semantics (CRSX). It will be shown how this framework supports abductive planning with Event Calculus ([4]) and some classical examples will be shown ([7, 11]) in the domain of information-seeking dialogues. Finally, some open problems and future work will be pointed out.

1 Introduction

A robust man-machine interaction requires the capability for inferring the beliefs, intentions and plans of each active agent.

In order to deal with these problems there has been done some work with different approaches. One major approach follows the classical planning scheme developed in the STRIPS ([5]) and NOAH ([13]) model. In this model each plan

is defined as a sequence of actions and each action is composed by an head, pre-conditions, constraints, effects and sub-actions. The inference of plans (a list of user actions) is done through the use of a library of plans and actions, some heuristic rules and the user possible goals. This approach has been used by Litman and Allen ([7, 6]) in order to infer plans behind speech acts in dialogues. A different approach was followed by Pollack ([11, 1]) which models plan as mental states and tries to abduct the mental attitudes behind each speach act.

In this paper we follow a general approach which will allow us to handle both models. Since we needed non-monotonic reasoning, namely default and abductive reasoning, as the basic inference process we have used the event calculus to represent events, time and actions and a logic programming framework with a given and defined semantics, Well Founded Semantics of eXtended Logic Programs (WFSX) augmented with Contradiction Removal Semantics (CRSX) from the work of Pereira ([10]). This framework extends logic programming and allows the modeling of several kinds of non-monotonic reasoning, namely dafault, abductive and hypothetical reasoning. Furthermore, it allows the removal of contradictions allowing the desambiguation of dialogues.

In section 2 a description of the framework showing how non-monotonic reasoning is dealt with is given. In section 3 the process of abductive planning with event calculus is described and in section 4 it is shown how this framework is able to handle the same kind of problems that Litman and Pollack handle. Finally in section 5 some open problems and future work will be pointed out.

2 Logic Programming Framework

In order to reason about plans and attitudes we need to model actions, events, states and world knowledge. In this framework they are modeled by extended logic programs which are a set of rules and integrity rules having the form:

- $H \leftarrow B_1, \ldots, B_n, notC_1, \ldots, notC_m (m \geq 0, n \geq 0)$

where H, B_1, ..., B_n, C_1, C_m are classical literals. A classical literal is either an atom A or its explicit negation $\neg A$. *not* stands for the negation by failure (NAF). In integrity rules H is the symbol $\bot$ (contradiction).

Default reasoning can be modeled adding to the program rules of the form:

- Normally $A(X)$ implies $B(X)$

which can be written as:

1. B(X) $\leftarrow$ A(X), not ab(X)

which states that if it's not possible to prove the abnormality of X then B should hold.

With a slight change it is possible to handle hypothetical reasoning:

- Quakers might (or not) be pacifists

which can be written as:

1. pacifist(X) ← quaker(X), hypqp(X)
2. hypqp(X) ← not ¬ hypqp(X)
3. ¬ hypqp(X) ← not hypqp(X)

which states that quakers are pacifists if it's not possible to prove (by NAF) explicitly that they are not (and vice-versa).

Abductive reasoning is modeled with rules:

- F might be true or not

which can be written as:

1. $F \leftarrow not \neg F$
2. $\neg F \leftarrow not F$

which state that if it's not possible to prove $\neg F$ then F should hold (and vice-versa).

Using this approach it's possible to create an abductive program from an abductive theory (P, Ab) by adding to the program P for all literals L in the abductible list Ab two rules of the form:

1. $L \leftarrow not \neg L$
2. $\neg L \leftarrow not L$

3 Abductive Planning with Event Calculus

In order to represent and reason about actions and events it was used the event calculus with some changes proposed by Eshgi ([4]) and Missiaen ([9]). The following logic program is proposed by Missiaen in order to describe what properties hold at a given time:

$$
\begin{aligned}
holds\_at(P,T) &\leftarrow happens(E), initiates(E,P), && (1)\\
&\quad succeeds(E), E < T, persists(E,P,T).\\
persists(E,P,T) &\leftarrow notclipped(E,P,T). && (2)\\
clipped(E,P,T) &\leftarrow happens(C), terminates(C,P), && (3)\\
&\quad succeeds(C), notout(C,E,T).\\
out(C,E,T) &\leftarrow (T = C; T < C; C < E). && (4)
\end{aligned}
$$

which states that the property P holds at time T if there was an event that happened before T, if that event initiates P and if P persists until T. A property P persists until T if it is not possible to prove (by NAF) the existence of an event that terminates the property P before T.

In order to infer the user plans it's necessary to abduct events (related with actions) and their temporal ontology. The set of abductible predicates in our framework is composed by:

Ab = {happens/1, act/2, </2}

This set if abductibles allow the framework to abduct the events that explain the observed effects.

The representation of actions in this framework can be done by a translation mechanism that for each action described by the statement:

- A causes F if P_1, ..., P_n

meaning that action A causes the effect F if the pre-conditions P_1, ..., P_n hold, produces the following rules:

1. succeeds(E) $\leftarrow$ act(E, A), holds-at(P_1, E), ..., holds-at(P_n, E).
2. initiates(E, F) $\leftarrow$ act(E, A).

This scheme states that an event E associated with an action A at a time T succeeds if the pre-conditions hold at that time and as a consequence of this event the property F will hold in the future.

4 Example

In the following two sections it will be presented two examples solved with the unified extended logic framework proposed. In the first example there is missing information that was not conveyed by the user. In the second example it will be pointed out how to solve dialogue situations where incorrect knowledge is assumed by the speaker and how the other agent can handle the situation.

4.1 Train Dialogue

Suppose the following example (from [7]):

- Passenger: The eighty-fifty to Montreal?
- Clerk: Eighty-fifty to Montreal? Gate Seven.

- Passenger: Where is it?
- Clerk: Down this way to the left. Second one on the left.
- Passenger: Ok.

In order to deal with this example we have to define the library of actions (domain actions, speech acts) needed to make inferences. We have used the actions defined in [7] and the translation mechanism described in the previous section.

Namely the following actions were defined:

- First rule:

goto(agent, location, time) *causes* at(agent, location, time).

This rule means that if an agent goes to a specific place at a given time then he will be at that place.

- Second rule:

meet(agent, arriveTrain) *if* goto(agent, gate(arriveTrain), time(arriveTrain)).

This rule means that if an agent goes to a specific gate of an arrival train at the arrival time then we will meet that train.

- Third rule:

board(agent, departTrain) *if*

goto(agent, gate(departTrain), time(departTrain)), geton(agent, departTrain).

This rule means that if an agent goes to a gate of a departure train and gets on the train then we will board the train.

- Forth rule:

take-train-trip(agent, departTrain, destination) *if*

buy-ticket(agent, clerk, ticket), board(agent, departTrain).

This rule means that an agent takes a train trip if he buys a ticket to that trip and boards the departing train.

It was also needed to define rules for some speech acts:

- First rule:

inform(speaker, hearer, proposition) *causes*

know(hearer, proposition),

know(hearer, know(speaker, proposition))

if

know(speaker, proposition),

surface-inform(speaker, hearer, proposition).

If a speaker knows a specific proposition and informs the hearer about that proposition then the hearer will learn the proposition and he will know that the speaker knows it.

- Second rule:

informref(speaker, hearer, term, proposition) *causes*

knowref(hearer, term, proposition),

if

knowref(speaker, term, proposition),

know(hearer, proposition),

parameter(term, proposition).

If a speaker knows about a property of a proposition then, if he informs the hearer about that property, the hearer will learn that fact.

- Third rule:

request(speaker, hearer, action) *causes*

want(hearer, action),

know(hearer, want(speaker, action)),

if

want(speaker, action),

(surface-request(speaker, hearer, action) or

surface-request(speaker, hearer, informif(hearer, speaker, cando(hearer, action))
or

surface-inform(speaker, hearer, ¬ cando(speaker, action)) or

surface-inform(speaker, hearer, want(speaker, action))).

If a speaker wants an action to be done he can perform one of the following four speech acts:

- Request the hearer to do the action;
- Request the hearer to inform him if the hearer is able to do the action;
- Inform the hearer that he is not able to do the action;
- Inform the hearer that he wishes the action to be done.

causing the hearer to want to do the action (it is a cooperative dialogue) and to know that the hearer wants the action to be done.

There were also defined meta-actions, allowing the inference of meta-plans and allowing the system to reason about the structure of the dialogue: *continue-plan*, *identify-parameter*, *correct-plan* and *modify-plan* ([7, 6]).

As an example of the translation process into extended logic programs, it is shown the result of this process for the *request* speech act:

$$\begin{aligned} suceeds(E) \leftarrow\ & act(E, request(s,h,a)), \\ & holds\_at(want(s,a),E), \\ & (holds\_at(surface\_request(s,h,a),E) or \\ & holds\_at(surface\_request(s,h,informif(h,s,cando(h,a)),E) or \\ & holds\_at(surface\_inform(s,h,\neg cando(s,a)),E) or \\ & holds\_at(surface\_inform(s,h,want(s,a)),E)). \end{aligned} \tag{5}$$

$$initiates(E, want(h,a)) \leftarrow act(E, request(s,h,a)). \tag{6}$$

$$initiates(E, know(h, want(s,a)) \leftarrow act(E, request(s,h,a)). \tag{7}$$

After this process the first user utterance creates the following facts:

1. happens(e1)
2. act(e1, request(passenger, clerk, informref(clerk, passenger, T, train(montreal))))
3. holds-at(knowref(passenger, time(8:50), train(montreal)), e1)

describing that the passenger requested to be informed by the clerk about a given reference of the 8:50 Montreal train.

It is possible to make inferences about the present, past and future situations. For instance it can be inferred what properties hold as a consequence of the speech act event:

1. holds-at(X, e1)
2. X = want(clerk, informref(clerk, passenger, T, tr(montreal))) and
3. X = know(clerk, want(passenger, informref(clerk, passenger, T, tr(montreal))))

Accordingly with the domain acts library and the speech acts defined by Litman it is possible to infer that the clerk wants to inform the passenger and that he knows that the passenger wants to be informed.

On the other hand it is possible to abduct what will hold as a consequence of this event (accordingly with the plan library):

1. happens(e2), e1 < e2,
2. holds-at(X, e2), act(e2, A)
3. A = informref(clerk, passenger, gate(7), tr(montreal))
4. X = know(passenger, gate(7), tr(montreal))

It is abducted that the clerk will inform the passenger about the missing property (it is a cooperative information-seeking dialogue).

Furthermore it's possible to go deeper and to infer what might happen next:

1. happens(e2), e1 < e2,
2. happens(e3), e2 < e3,
3. holds-at(X, e3), act(e3, A)
4. A = goto(passenger, gate(7))
5. X = at(passenger, gate(7))

In this situation it's possible that the passenger will go to the Montreal gate.

If this process is taken deeper, it's possible to abduct that the passenger will go to the gate in order to take a train trip (the desambiguation between meeting and boarding can be done through the access of a train knowledge base with information about trains, departure and arrival hours and destinations) and that the action is a first step of the meta-plan *introduce-plan*.

4.2 Computer mail messages

In these examples (adapted from [11]) problems with dialogues with error situations are pointed out. In the first example the user plan is incorrect in order to achieve the user goal while in the second example it's the user goal that is impossible to be achieved.

Example 1:

- Q: I want to talk to Kathy? Do you know the phone number at the hospital?
- A: She's already been discharged. Her home number is 555-8321.

Example 2:

- Q: How can I protect my mail messages? I don't want the system manager to be able to read them.
- A: Even if you set the protections, the system manager can override them.

In these examples, the plan ascription is made through the inference of the agents beliefs and intentions. As epistemic operators (describing the agents mental states) we've defined the following ([1, 3]):

INT(a, α): agent a intends to do α

BEL(a, p): agent a beliefs that p is true

ACH(a, p):agent a beliefs p will be true as a consequence of its actions

EXP(a, p) $\leftarrow$ BEL(a, p) or ACH(a, p)

More complex actions can be constructed from the operators TO e BY:

TO(α, p): the plan of performing α in order to make p true

BY(α, β, p): the plan of making β by doing α, while p is true

It's also needed to define some rules that connect these epistemic operators:

INT(a, TO(α, p)) $\leftarrow$ BEL(a, TO(α, p)), INT(a, α), ACH(a, p)

These rule means that if an agent beliefs that doing α makes p true, if he intends to do α and if he wants p to become true, then he intends to do α in order to make p true.

There is also the corresponding rule for the relation BY:

INT(a, BY(α, β, p)) $\leftarrow$ BEL(a, BY(α, β, p)), INT(a, α), INT(a, β), EXP(a, p)

This rule means that if an agent beliefs that by doing α β is done and if he intends to do α and β and if he expects p to be true, then he intends to do α in order to do β while p is true.

It's also needed an integrity constraint, showing the inconsistency between BEL e ACH:

$\perp \leftarrow$ BEL(a, p), ACH(a, p)

In fact, it's impossible to believe simultaneously that p is true and to wish p to become true.

Using these rules and the Event Calculus it's possible to reason about the examples previously presented. In the first example the user question might create the following facts:

1. happens(e1)
2. act(e1, request(user, system, infref(system, user, number(N), hospital)))
3. initiates(e1, int(user, talk(kathy)))

With these facts and using an abductive inference over the actions and events it's possible to infer that:

holds-at(e1, int(user, BY(phone(hospital), talk(kathy), at-hospital(kathy))))

The user intention by phoning to the hospital is to talk to Kathy (assuming she is at the hospital). In order to make this inference there were abducted the following facts:

1. exp(user, at-hospital(kathy))
2. bel(user, by(phone(hospital), talk(kathy), at-hospital(kathy)))

connected with an event $e1$.

If the system knows that Kathy is at home then the inference is incorrect (from the system's point of view), and it's possible the system to infer:

1. exp(system, at-home(kathy))
2. holds-at(e1, bel(system, by(home(hospital), talk(kathy), at-home(kathy))))

With this process it's possible to generate by the system an answer similar to the one in the first example.

In the second example, the user question can generate the following facts:

- happens(e1)

- act(e1, request(user, system, infref(system, user, protect(N), mail)))
- initiates(e1, int(user, ¬ read(sm, mail)))

In this situation it's impossible to achieve the desired goal using the rules and facts known. There is no causal relation between the speech act and the user goal. The system might inform the user about the impossibility of reaching the goal (with an adequate explanation).

5 Conclusions and Future Work

We have sketchly presented in this paper a framework that supports the abductive inference of actions and events through the use of extended logic programs and the event calculus. This framework allows the implementation of a contradiction removal mechanism that is able to remove possible contradictions from the logic program. Using this mechanism it's also possible to handle ambiguous situations in dialogues and to choose the best interpretation for the utterances.

The results obtained show that this framework is able to handle some of the problems that arise in dialogues (non-specified goals, clarification sub-dialogues, error situations) and it can support the inference of plans and attitudes in a more general natural language processing system such as the one described by Lopes and Quaresma ([8, 12]) where a multi-headed architecture coordinates several independent modules with the shared objective of supporting a robust natural language interaction. It will be necessary to evaluate comparatively the results obtained by Cohen in Levesque ([2]).

References

[1] Douglas E. Appelt and Martha E. Pollack. Weighted abduction for plan ascription. *User Modeling and User-Adapted Interaction*, 2(1), 1992.

[2] P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.

[3] P. Cohen, J. Morgan, and M. Pollack. *Intentions in Communication.* MIT Press, Cambridge, MA, 1990.

[4] Kave Eshghi. Abductive planning with event calculus. In *Proceedings of the International Conference on Logic Programming*, 1988.

[5] R. E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intellligence*, (2):189–208, 1971.

[6] D. Litman and J. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, (11):163–200, 1987.

[7] Diane J. Litman. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*. PhD thesis, Dep. of Computer Science, University of Rochester, 1985.

[8] J. G. Lopes. Architecture for intentional participation of natural language interfaces in conversations. In C. Brown and G. Koch, editors, *Natural Language Understanding and Logic Programming III*. North Holland, 1991.

[9] Lode Missiaen. *Localized Abductive Planning with the Event Calculus*. PhD thesis, Univ. Leuven, 1991.

[10] Luís Moniz Pereira, José Júlio Alferes, and Joaquim Nunes Aparício. Contradiction removal semantics with explicit negation. In M. Masuch and L. Pólos, editors, *Knowledge Representation and Reasoning Under Uncertainty, Volume 808 of LNAI*, pages 91–106. Springer-Verlag, 1994.

[11] Martha E. Pollack. *Inferring Domain Plans in Question-Answering*. PhD thesis, Dep. of Computer and Information Science, University of Pennsylvania, 1986.

[12] P. Quaresma and J. G. Lopes. A two-headed architecture for intelligent multimedia man-machine interaction. In *B. de Boulay and V. Sgurev (eds). Artificial Intelligence V - methodology, systems, applications*. North Holland, 1992.

[13] Earl D. Sacerdoti. *A Structure for Plans and Behavior*. American Elsevier, New York, 1977.