UNIFIED LOGIC PROGRAMMING APPROACH TO THE ABDUCTION OF PLANS AND INTENTIONS IN INFORMATION-SEEKING DIALOGUES *

PAULO QUARESMA AND JOSÉ GABRIEL LOPES

▷ We propose a framework that supports the recognition of plans and intentions behind speech acts through abductive inferences over discourse sentences. These inferences allow each agent to have an active and intelligent participation in dialogues, namely, in cooperative information-seeking dialogues. In our framework, the possible actions, events, states, and world knowledge are represented by extended logic programs (LP with explicit negation), and the abductive inference process is modeled by the framework proposed by Pereira et al. [13], 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 [5], and some examples will be shown [10, 14] in the domain of information-seeking dialogues. Finally, some open problems will be pointed out.

*This work has been funded by JNICT under Contract IDILIO. The work of P. Quaresma was supported by a JNICT scholarship reference no. BD/1766/IA.

Address correspondence to Paulo Quaresma, Artificial Intelligence Center, UNINOVA, 2825 Monte da Caparica, Portugal. E-mail: {pq,gpl}@fct.unl.pt.

Received September 1993; accepted January 1995.

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 a lot of work done taking different views and approaches. One major approach follows the classical planning scheme developed in the STRIPS [6] and NOAH [18] model. In this model each plan is defined as a sequence of actions and each action is composed by a head, preconditions, constraints, effects, and subactions. The inference of plans (a list of user actions) is achieved by using a library of plans and actions, some heuristic rules, and the user possible goals. This approach has been used by Litman and Allen [10, 9] in order to infer plans behind speech acts in dialogues. A different approach was followed by Pollack [14, 2] which models plans as mental states and tries to abduce the mental attitudes behind each speech act.

We propose a general extended logic programming framework which will allow us to handle both models and to deal with the same kind of dialogues that are dealt with in their approaches. The basic point is that the inference of plans, intentions, and beliefs supporting speech acts during dialogues is a nonmonotonic process. For instance, in the following dialogue (from the train station domain), the clerk needs to revise his beliefs about the passenger intentions (after the passenger's second utterance).

- Passenger: The eight-fifty train?
- Clerk: Eight-fifty to Porto? Gate Seven.
- Passenger: No, the eight-fifty from Porto.
- Clerk: Oh, it will arrive about ten minutes late at gate five!
- Passenger: Ok.

Since we need, as the basic inference process, nonmonotonic reasoning, we have used the event calculus to represent events, time, and actions and a logic programming framework – Well Founded Semantics of eXtended Logic Programs (WFSX) augmented with Contradiction Removal Semantics (CRSX) – from the work by Pereira et al. [13]. This framework has a given and defined semantics and extends logic programming. It enables one to model several kinds of nonmonotonic reasoning, namely, default, abductive, and hypothetical reasoning.

In Section 2, a description of the extended logic programming framework showing how nonmonotonic reasoning is dealt with is given. In Section 3, the process of abductive planning with event calculus is described. Section 4 describes the language that was used to represent actions (from Gelfond and Lifschitz [7]). Section 5 describes the epistemic operators used to model the different attitudes. In Section 6, the speech acts necessary to handle dialogues are described and in Section 7, the rules that define the agents' behavior are described. In Section 8, it is shown how this framework is able to handle some classical problems in dialogue understanding: lack of information in the utterances, and wrong user plans. Finally, in Section 9, some open problems and future work will be pointed out.

2. EXTENDED LOGIC PROGRAMMING FRAMEWORK

As was pointed out in Section 1, we need nonmonotonic reasoning in order to reason about plans and attitudes, and we have to model actions, events, states, and world knowledge. In this framework, they are modeled by extended logic programs which are sets of normal and integrity rules having the form

- $H \leftarrow B_1, \ldots, B_n, \text{not } C_1, \ldots, \text{not } C_m (m \geq 0, n \geq 0)$

where H, B_1, $\ldots$, 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 as failure (NAF). In integrity rules, H is the symbol $\perp$ (contradiction).

Programs with integrity constraints and explicit negation may be contradictory. For instance, in the following program $P = \{a \leftarrow \text{not } b, \perp \leftarrow a\}$ (from [13]), *not b* is true by Closed World Assumption and, by the second rule, we have a contradiction.

The framework proposed by Pereira et al. removes the contradiction by adding to the original program the complements of some revisables. In the previous example, it would add a rule to prevent the CWA over the first rule. The minimal set of revising assumptions of a program P is calculated through the use of an iterative algorithm. It was shown that this algorithm is sound and complete for a finite set of revisables.

The framework described allows the modeling of different types of nonmonotonic reasoning, namely, default, hypothetical, and abductive reasoning. Nonmonotonic reasoning is fundamental for the inference of plans and intentions in dialogues, as will be shown in Section 8.

- 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) ← A(X), not ab(X)

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

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

Quakers may or may 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 is not possible to prove (by NAF) explicitly that they are not (and vice versa).

- Abductive reasoning is modeled with the 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 is not possible to prove $\neg F$, then F should hold (and vice versa).

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

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

In the domain of dialogues, we will allow events to be abduced. This process is done through the use of this framework and the Event Calculus.

3. ABDUCTIVE PLANNING WITH EVENT CALCULUS

In order to represent and reason about actions and events we use the event calculus with some changes proposed by Eshgi [5] and Missiaen [12].

In this formalism, events are action instances and have no duration. They can be represented by the time point at which they occurred, and time points are ordered by the precedence relationship $<$. Events are related to its actions by the predicate $act(E, A)$. $happens(E)$ states that the event E occurs. $initiates(E, P)$ means that the event E initiates the property P, and $terminates(E, P)$ states that property P is terminated by the event E. $succeeds(E)$ means that the event E can happen, i.e., its preconditions are satisfied (different from *happens*, which means that the event really happened). $holds\_at(P, T)$ means that property P holds at time T.

A plan consists of the set of facts defined by the predicates $<$, *happens*, and *act*. This set can represent a plan because it defines the events that happen, the actions that are associated with those events, and the time ordering between the events.

There are some properties about plans that can be defined, namely:

A plan is partial if there is a partial ordering between the time points of the plan. It is linear if its time points are totally ordered. A total plan is a linearization of a given plan that satisfies the partial order of that plan.

A plan P is a solution for a goal G wrt the theory T if

1. There exists a substitution σ such that $T + P \vdash \sigma(G)$, where $+$ states the union of sets.
2. Every event of P succeeds, i.e.,

$$\forall_{e_i} happens(e_i) \in P, T + P \vdash succeeds(e_i).$$

Finally, a plan is valid if, for every variable substitution for which the plan is a solution, the linearizations of the plan are also solutions (for the same variable substitution).

The following logic program is proposed by Missiaen in order to describe what properties hold at a given time:

$$\begin{aligned}
holds\_at(P,T) &\leftarrow happens(E), initiates(E,P), \\
&\quad\; succeeds(E), E < T, persists(E,P,T). && (1)\\
persists(E,P,T) &\leftarrow not\ clipped(E,P,T). && (2)\\
clipped(E,P,T) &\leftarrow happens(C), terminates(C,P), \\
&\quad\; succeeds(C), not\ out(C,E,T). && (3)\\
out(C,E,T) &\leftarrow (T = C; T < C; C < E). && (4)
\end{aligned}$$

which states that the property P holds at time T if there was an event that occurred 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. Note the importance of the predicate *succeeds*(E) forcing the preconditions of the event E to be satisfied.

This logic program is different from the one proposed by Shanahan [19] because, in Shanahan's work, if C cannot be proved to be inside $[E, T[$, then it is assumed to be outside, while in Missien's work, if C cannot be proved to be outside the interval, then it is assumed to be inside. This implies that the programs have different completions when there is a partial order between the events. If the events have a total order between them, then the two programs are equivalent.

We have chosen the Missien proposal because, in this approach, every solution plan is a valid one, i.e., all of its linearizations are correct, while in Shanahan's approach, it is not guaranteed that every solution plan is valid.

However, it is still an open problem to prove that this approach is correct, complete, and/or sound wrt. the well founded semantics framework described in the previous section.

As was pointed out earlier, in order to infer a user's plan, it is necessary to find the plans that satisfy the following implication:

- $T + P \vdash \sigma(G)$.

In this approach, planning can be seen as an abductive process in which the set of abducible predicates is composed of

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

This set of abducibles allows the framework to abduce events (related with actions) and their temporal relations.

4. REPRESENTING ACTIONS

In order to represent actions, we have used the language proposed by Gelfond and Lifschitz [7]. In this language, actions can be represented by the following two propositions:

$$\begin{array}{lll} A & causes & F\ if\ P_1, \ldots, P_n \\ F & after & A_1, \ldots, A_n. \end{array}$$

The first rule means that action A causes the effect F if the preconditions P_1, ..., P_n hold, and the second rule states that the fluent F will hold as a consequence of the sequence of actions $A_1, \ldots, A_n$.

In this work, we are only using the first rule, but we allow rules without actions (F if $P_1, \ldots, P_n$) and actions without consequences (A if $P_1, \ldots, P_n$).

We have defined a translation mechanism from this language into the event calculus language introduced earlier. For each action described by the first statement, the following two rules are obtained:

$$\begin{array}{rcl} succeeds(E) & \leftarrow & act(E,A), holds\_at(P_1,E), \ldots, holds\_at(P_n,E). \\ initiates(E,F) & \leftarrow & act(E,A), holds\_at(P_1,E), \ldots, holds\_at(P_n,E). \end{array}$$

This scheme states that an event E associated with an action A at a time E is possible if the preconditions hold at that time, and as a consequence of the event, the property F will hold in the future. Note that the preconditions are also verified in the second rule because action A can have different effects when different conditions hold. For example, shooting a gun causes damages only if the gun is loaded.

If the action has no direct effects, the rule can be written as

$$A \quad if \quad P_1, \ldots, P_n$$

and translated into

$$succeeds(E) \quad \leftarrow \quad act(E,A), holds\_at(P_1,E), \ldots, holds\_at(P_n,E).$$

If we want to define a rule which relates only fluents, we can write the following rule:

$$F \quad if \quad P_1, \ldots, P_n$$

which will be translated into

$$holds\_at(F, E) \quad \leftarrow holds\_at(P_1, E), \ldots, holds\_at(P_n, E).$$

Note that the operator if is overloaded in the situations F if $P_1, \ldots, P_n$ and A if $P_1, \ldots, P_n$. This is not a problem, and it can be solved because the left operands are different in the two stated cases.

On the other hand, it is very important to point out that we have not proved that the proposed transformations into the abductive event calculus are correct and complete.

In order to apply the described framework to a specific domain, we have to translate the domain specific rules defined using this language into extended logical rules. For instance, the blocks world action

$$move\_to\_top\_of(X, Y) \quad causes \quad on(X, Y) \; if \; on\_hand(X), free(Y).$$

is translated into the rules

$$\begin{aligned} succeeds(E) \quad &\leftarrow \quad act(E, move\_to\_top\_of(X, Y)), \\ &\qquad holds\_at(on\_hand(X, Y), E), holds\_at(free(Y), E). \\ initiates(E, on(X, Y)) \quad &\leftarrow \quad act(E, move\_to\_top\_of(X, Y)), \\ &\qquad holds\_at(on\_hand(X, Y), E), holds\_at(free(Y), E). \end{aligned}$$

5. EPISTEMIC OPERATORS

As epistemic operators needed to describe the agents' mental state, we have defined the following [2, 4]:

1. int(a, α): agent a wants action α to be done
2. bel(a, p): agent a believes that p is currently true
3. ach(a, p):agent a believes p will be true as a consequence of the actions of some agent (including its own actions)
4. exp(a, p) $\leftarrow$ bel(a, p) or ach(a, p): agent a expects the fluent p to be or to become true.

More complex actions can be constructed from the operators *to* and *by*:

1. to(α, p): the plan of performing α in order to make p true
2. by(α, β, p): the plan of making β by doing α, while p is true

For example, the inform act

- I can't open the door.

could be a step of the plan to request the hearer to open the door:

$$by(inform(s, h, cantdo(s, open\_door(s))), request(s, h, open\_door(h)), ach(s, door\_opened)).$$

And this request could be a step of the complex action

$$to(request(s, h, open\_door(h)), door\_opened).$$

It was also necessary to define some rules that connect these epistemic operators:

$$\begin{array}{lll} int(A, to(\alpha, P)) & if & bel(A, to(\alpha, P)) \\ & & int(A, \alpha), \\ & & ach(A, P). \end{array}$$

This rule means that if an agent A believes that by doing α P will become true, and A intends to do α and A wants P to become true, then A intends to do α in order to make P true.

There is also the corresponding rule for the relation *by*:

$$\begin{array}{lll} int(A, by(\alpha, \beta, P)) & if & bel(A, by(\alpha, \beta, P)), \\ & & int(A, \alpha), \\ & & int(A, \beta), \\ & & exp(A, P). \end{array}$$

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

These rules are translated into the following logical rules:

$$\begin{array}{lll} holds\_at(int(A, to(\alpha, P)), E) & \leftarrow & holds\_at(bel(A, to(\alpha, P)), E), \\ & & holds\_at(int(A, \alpha), E), \\ & & holds\_at(ach(A, P), E). \end{array} \tag{5}$$

$$\begin{array}{lll} holds\_at(int(A, by(\alpha, \beta, P)), E) & \leftarrow & holds\_at(bel(A, by(\alpha, \beta, P)), E), \\ & & holds\_at(int(A, \alpha), E), holds\_at(int(A, \beta), E), \\ & & holds\_at(exp(A, P), E). \end{array} \tag{6}$$

Also needed is an integrity constraint, showing the inconsistency between *bel* and *ach*:

$$\perp \;\leftarrow\; bel(A,P), ach(A,P). \tag{7}$$

In fact, it is impossible to believe that P is true and simultaneously to wish P to become true.

Finally, we have two more rules:

$$\begin{array}{lll} bel(A, int(A,\alpha)) & if & int(A,\alpha) \\ bel(A, bel(A,P)) & if & bel(A,P). \end{array}$$

These rules mean that an agent believes in his intentions and beliefs. They are translated into

$$holds\_at(bel(A, int(A,\alpha)), E) \;\leftarrow\; holds\_at(int(A,\alpha), E). \tag{8}$$

$$holds\_at(bel(A, bel(A,P)), E) \;\leftarrow\; holds\_at(bel(A,P), E). \tag{9}$$

6. SPEECH ACTS

Using the language described in the previous section we have defined expressions for some speech acts (from [1]):

1. inform(s, h, p) : s informs h about the proposition p
2. informref(s, h, t, p) : s informs h about the term t of the proposition p
3. request(s, h, a) : s requests h to do action a.

The first action is described by

$$\begin{array}{lll} inform(s,h,p) & causes & bel(h, bel(s,p)) \\ & if & bel(s,p) \\ & & bel(s, int(s, inform(s,h,p))). \end{array}$$

If a speaker believes in a specific proposition and intends to inform the hearer about that proposition, then the hearer will believe that the speaker believes it. This rule models a naive behavior where the agents only inform propositions in which they believe.

The translation into logical rules is

$$\begin{aligned} succeeds(E) &\leftarrow act(E, inform(S,H,P)),\\ &\quad holds\_at(bel(S,P),E)\\ &\quad holds\_at(bel(S,int(S,\\ &\qquad inform(S,H,P))),E). \qquad (10)\\ initiates(E, bel(H, bel(S,P))) &\leftarrow act(E, inform(S,H,P)),\\ &\quad holds\_at(bel(S,P),E),\\ &\quad holds\_at(bel(S,\\ &\qquad int(S, inform(S,H,P))),E). \qquad (11) \end{aligned}$$

The second action is defined by

$$\begin{array}{lll} informref(s,h,t,p) & causes & bel(h, bel(s, ref(t,p)))\\ & if & bel(s, ref(t,p))\\ & & bel(s, int(s, informref(s,h,t,p). \end{array}$$

If a speaker believes t is a feature value pair of p and the speaker intends to inform the hearer about that property, then the hearer will believe that the speaker believes that t is a property of p. The logical rules are

$$\begin{aligned} succeeds(E) &\leftarrow act(E, informref(S,H,T,P)),\\ &\quad holds\_at(bel(S, ref(T,P)),E)\\ &\quad holds\_at(bel(S,int(S,\\ &\qquad informref(S,H,T,P))),E). \qquad (12)\\ initiates(E, bel(H, bel(S, ref(T,P)))) &\leftarrow act(E, informref(S,H,T,P)),\\ &\quad holds\_at(bel(S, ref(T,P)),E)\\ &\quad holds\_at(bel(S,int(S,\\ &\qquad informref(S,H,T,P))),E). \qquad (13) \end{aligned}$$

The third speech act can be defined by

$$\begin{array}{lll} request(s,h,a) & causes & bel(h, bel(s, int(s,a)))\\ & if & bel(s, cando(h,a))\\ & & bel(s, int(s, request(s,h,a))). \end{array}$$

If a speaker S believes a hearer H can do an action A, and the speaker believes that he wants to request the hearer to do the action, then he may actually do the request.

Note that the request for an action has the same informational effects as a desire. For example, the sentences

- Can you open the door?

- I want you to open the door.

will have as an effect the following proposition:

$$bel(H, bel(S, int(S, open(H, door)))).$$

The request expression is translated into

$$\begin{aligned} succeeds(E) &\leftarrow act(E, request(S, H, A)), \\ &\quad holds\_at(bel(S, cando(H, A)), E) \\ &\quad holds\_at(bel(S, int(S, \\ &\qquad request(S, H, A))), E). \qquad (14) \\ initiates(E, bel(H, bel(S, int(S, A)))) &\leftarrow act(E, request(S, H, A)), \\ &\quad holds\_at(bel(S, cando(H, A)), E) \\ &\quad holds\_at(bel(S, int(S, \\ &\qquad request(S, H, A))), E). \qquad (15) \end{aligned}$$

7. AGENTS' MENTAL STATES

We have defined two rules that define the mental states of the different agents that participate in the dialogues.
The first rule defines the degree of cooperativeness of a dialogue:

$$\begin{aligned} bel(h, int(h, a)) \quad if \quad & bel(h, bel(s, int(s, a))), \\ & bel(h, cando(h, a)). \end{aligned}$$

This means that if an agent h believes that the agent s wants action a to be performed and h believes he can do that action then he will believe he intends to perform it. The dialogues are totally cooperative.
The rule is translated into

$$\begin{aligned} holds\_at(bel(H, int(H, A)), E) \leftarrow \; & holds\_at(bel(H, cando(H, A)), E) \\ & holds\_at(bel(H, bel(S, int(S, A))), E) \qquad (16) \end{aligned}$$

The second rule defines how an agent can be convinced by the other agents:

$$bel(h, p) \quad if \quad bel(h, bel(s, p)).$$

This rule means that the agent h believes in everything that he believes that the other agent believes. This approach is also a simplification of the reality and, in a more complex system, the agent would not believe in a proposition without first checking it.
The rule is translated into

$$holds\_at(bel(H, P), E) \leftarrow holds\_at(bel(H, bel(S, P)), E). \qquad (17)$$

8. EXAMPLES

In the following subsections, some examples solved and implemented in a prototype developed at the Artificial Intelligence Center of UNINOVA will be presented.

The first example is a dialogue in the train station domain where there is missing information in the passenger utterances. The clerk has to infer the intentions and the plans behind the passenger utterances.

In the second example, it will be pointed out how to solve dialogue situations where incorrect knowledge is assumed by the speaker. The other agent needs to infer what are the speaker beliefs that do not hold, and it has to infer the correct plan to achieve the desired goals.

8.1. Train Dialogue

Consider the example presented in Section 1 (adapted from [10]):

- Passenger: The eight-fifty train?
- Clerk: Eight-fifty to Porto? Gate Seven.
- Passenger: No, the eight-fifty from Porto.
- Clerk: Oh, it will arrive about ten minutes late at gate five!
- Passenger: Ok.

In order to deal with this example, we have used a simple knowledge base of information about trains, the speech acts defined in Section 6, and a library of domain actions needed to make inferences. The domain actions were adapted from [10], and the translation mechanism described in Section 4 was used.

8.1.1. Domain Actions. As knowledge base, we have a collection of facts defining the timetable for each train:

$$timetable(Train, Place, ArrivingTime, DepartingTime, Gate).$$

And some rules defining the clerk's beliefs:

$$\begin{aligned}
&holds\_at(bel(c, ref(from(Place), train(X))), E) \leftarrow \\
&\qquad timetable(X, Place, 0, DepartingTime, Gate). && (18)\\
&holds\_at(bel(c, ref(to(Place), train(X))), E) \leftarrow \\
&\qquad timetable(X, Place, ArrivingTime, 0, Gate). && (19)\\
&holds\_at(bel(c, ref(depart(Time), train(X))), E) \leftarrow \\
&\qquad timetable(X, Place, ArrivingTime, Time, Gate). \\
&\qquad at(c, Place). && (20)\\
&holds\_at(bel(c, ref(arrive(Time), train(X))), E) \leftarrow \\
&\qquad timetable(X, Place, Time, DepartingTime, Gate).
\end{aligned}$$

$$at(c, Place). \tag{21}$$

$$holds\_at(bel(c, ref(gate(Gate), train(X))), E) \leftarrow$$
$$timetable(X, Place, ArrivingTime, DepartingTime, Gate) \tag{22}$$

Note that an arriving/departing time of 0 means that the place is the starting/ending point of that train.

As an example, we could have the following knowledge base (we are assuming that the clerk is in Coimbra):

$$timetable(a, Lisboa, 0, 7{:}15, 4). \tag{23}$$
$$timetable(a, Coimbra, 8{:}45, 8{:}50, 7). \tag{24}$$
$$timetable(a, Porto, 10{:}15, 0, 3). \tag{25}$$
$$timetable(b, Porto, 0, 7{:}20, 2). \tag{26}$$
$$timetable(b, Coimbra, 8{:}50, 8{:}55, 5). \tag{27}$$
$$timetable(b, Lisboa, 10{:}20, 0, 2). \tag{28}$$
$$at(c, Coimbra). \tag{29}$$

As domain actions, we have defined the following actions:

1. goto(agent, location)
2. meet(agent, arriveTrain)
3. board(agent, departTrain)
4. take_train_trip(agent, departTrain, destination).

These actions were described using the representation language defined in Section 4:

$$goto(agent, location) \quad causes \quad at(agent, location).$$

This expression means that if an agent goes to a specific place, then he will be at that place.

The second action is defined by the expression

$$meet(agent, train) \quad if \quad at(agent, gate), at(train, gate).$$

This expression means that if an agent is at a gate, he meets the train that is there.

The third action is defined by

$$\begin{array}{lll} board(agent, train) & causes & at(agent, train) \\ & if & at(train, gate), at(agent, gate). \end{array}$$

This rule means that if an agent is at a departing train gate and boards on that train, then s/he will be in the train.

The last action is defined by the expression

$$\begin{aligned} take\_train\_trip(agent, departTrain, destination) \quad if \quad & has\_ticket(agent), \\ & at(agent, departTrain). \end{aligned}$$

This rule means that an agent takes a train trip if he has a ticket and he is in the departing train.

8.1.2. Generic Rules. Two generic rules that connect the speech acts and the domain actions were also defined.
The first rule states

$$\begin{aligned} informref(C, P, T, train(X)) \quad if \quad & bel(C, int(C, informref(C, P, T, train(X)))), \\ & bel(C, ref(T, train(X))), \\ & not\ bel(C, bel(P, ref(T, train(X)))), \\ & bel(C, bel(P, ref(T1, train(X)))), T1 \neq T. \end{aligned}$$

This expression states that if a clerk C intends to inform a passenger P about a known specific property T of a train X, and if s/he believes that the passenger does not know that property (but knows a different property), then the clerk can perform the inform action. Note that with this rule, the clerk only informs passengers about trains that they already know something about (destination, time, ...).
This expression is translated into the rule

$$\begin{aligned} succeeds(E) \quad \leftarrow \quad & act(E, informref(C, P, T, train(X))), \\ & holds\_at(bel(C, int(C, informref(C, P, T, train(X)))), E), \\ & holds\_at(bel(C, ref(T, train(X))), E), \\ & not\ holds\_at(bel(C, bel(P, ref(T, train(X)))), E), \\ & holds\_at(bel(C, bel(P, ref(T1, train(X)))), E), T1! = T. \ (30) \end{aligned}$$

The second rule states

$$\begin{aligned} bel(C, bel(P, ref(depart(T), train(X)))) \quad if \quad & bel(C, bel(P, ref(time(T), train(X)))), \\ & not\ bel(C, bel(P, ref(arrive(T), train(X)))). \end{aligned}$$

This expression states that a clerk may infer that a mentioned time is a departing time (if there is no evidence to believe it is an arriving time).
This expression is translated into the rule

$$\begin{aligned} & holds\_at(bel(C, bel(P, ref(depart(T), train(X)))), E) \leftarrow \\ & \quad holds\_at(bel(C, bel(P, ref(time(T), train(X)))), E), \\ & \quad not\ holds\_at(bel(C, bel(P, ref(arrive(T), train(X)))), E). \qquad (31) \end{aligned}$$

8.1.3. Inference Process. The first user utterance

- Passenger: The eight-fifty train?

creates the following facts:

$$\begin{array}{l} happens(e1). \\ act(e1, request(p, c, informref(c, p, P, train(X)))). \\ act(e1, inform(p, c, ref(time(8:50), train(X)))). \end{array}$$

describing that the passenger requested to be informed by the clerk about a given reference of a train and that s/he knows the time of that train. Note that, in this example, we are assuming that the clerk is only concerned with trains.

Moreover, in this paper we are not dealing with the transformation process between the natural language sentences and our framework.

If we pose the question about what are the clerk beliefs:

$$holds\_at(bel(c, X), e1) \tag{32}$$

the answer will be (using rule 1 and the request rule 15):

$$X = bel(p, int(p, informref(c, p, P, train(T)))). \tag{33}$$

This means that the clerk believes that the passenger wants to be informed about a specific property of a train. In order to infer this belief, it was necessary to abduce an event $e0 < e1$ that initiated the preconditions of the request rule (the passenger believes that the clerk can inform him about the reference of the train and, moreover, the passenger wants to request the clerk to do the $informref$ action).

Using rule 16 for cooperativeness, the previous fact, and abducing an event that initiates the fluent that the clerk can perform the action, we will have

$$X = int(c, informref(c, p, P, train(T))) \tag{34}$$

meaning that the clerk intends to perform the action.

On the other hand, using the second fact and the rule for the inform action (11), we will have

$$X = bel(p, ref(time(8:50), train(X))) \tag{35}$$

meaning that the clerk believes that the passenger is interested in an 8:50 train.

Using rule 31:

$$X = bel(p, ref(depart(8:50), train(X))) \tag{36}$$

meaning that the clerk believes that the passenger is interested in a train departing at 8:50 (there is no evidence to support that it is an arriving train).

Taking these beliefs into account, it is possible to plan the clerk's actions. In fact, the clerk has beliefs about the passenger intentions and has his own intentions to fulfill. One possible action would be to inform the passenger about the train gate (using rules 19, 22, 24, 25, and 30):

$$\begin{aligned} &informref(c, p, ref(to(porto), train(X))) \\ &informref(c, p, ref(gate(7), train(X))). \end{aligned} \tag{37}$$

Note that the clerk could inform the passenger about another unknown property (we have assumed an order of priorities between the properties).

On the other hand, if in the sequence of the dialogue the passenger clarifies the term 8:50 as an arriving time, it is possible to perform the same kind of inference and to perform an informref about the train arriving at 8:50 (as in the dialogue).

In this work, we are not dealing with the natural language generation of the different speech acts.

8.2. Computer Mail Messages

In the following example (adapted from [14]), the user plan is detected to be incorrect in order to achieve the stated goal.

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

It was necessary to define two domain basic rules that state some common sense reasoning:

$$\begin{aligned} bel(X, bel(A, by(call(A, L), talk(A, P), at(L, P)))) \quad if \quad & bel(X, int(A, talk(A, P))) \\ & bel(X, exp(A, at(L, P))). \end{aligned}$$

This rule means that if an agent believes that another agent intends to talk to someone that s/he expects to be at a specific place, then the agent believes that the other agent believes that by calling to that place s/he will talk to the person s/he is looking for. The rule is translated into

$$\begin{aligned} &holds\_at(bel(X, bel(A, by(call(A, L), talk(A, P), at(L, P)))), E) \leftarrow \\ &\quad holds\_at(bel(X, int(A, talk(A, P))), E), \\ &\quad holds\_at(bel(X, exp(A, at(L, P))), E). \end{aligned} \tag{38}$$

The second rule states that

$$\begin{aligned} bel(A, int(B, informref(A, B, phone\_number(N), L))) \quad if \quad & bel(A, int(B, talk(B, P))) \\ & bel(A, exp(B, at(L, P))), \\ & \neg bel(A, \\ & \quad know(B, phone(N, L))). \end{aligned}$$

This rule means that an agent A believes the other agent B wants to be informed about the phone number N of a specific place L if the agent A believes that the other agent intends to talk to a person that he expects to be at the place L. Moreover, A believes that B does not know the correct phone number. The rule is translated into

$$\begin{aligned} & holds\_at(bel(A, int(B, informref(A, B, phone\_number(N), L))), E) \leftarrow \\ & \quad holds\_at(bel(A, int(B, talk(B, P))), E), \\ & \quad holds\_at(bel(A, exp(B, at(L, P))), E), \\ & \quad \neg holds\_at(bel(A, know(B, phone\_number(N, L))), E). \end{aligned} \tag{39}$$

In the first example, the user question might create the following facts:

$$\begin{aligned} & happens(e0), \\ & act(e0, inform(user, system, int(user, talk(user, kathy)))), \\ & happens(e1), e0 < e1, \\ & act(e1, request(user, system, \\ & \quad informref(system, user, phone\_number(N), hospital))) \end{aligned} \tag{40}$$

If we pose the question about the system's beliefs:

$$holds\_at(bel(system, X), e1)$$

using rule 1, the $inform$ rule 10, and action $e0$, ww will have

$$X = bel(user, int(user, talk(user, kathy)))$$

and using rule 17:

$$X = int(user, talk(user, kathy))$$

This means that the system believes that the user wants to talk to Kathy.

On the other hand, using the request rule 15 and the action $e1$, we will have

$$X = bel(user, int(user, informref(system, user, phone\_number(N), hospital)))$$

meaning that the system believes the user believes that he wants to be informed by the system about the phone number of the hospital.

Using rule 17:

$$X = int(user, informref(system, user, phone\_number(N), hospital))$$

meaning that the system believes the user wants to be informed by the system about the phone number of the hospital.

With this fact and abducing an event that initiates the preconditions of rule 39 (the fluents *bel(system, exp(user, at(hospital, kathy)))* and ¬ *bel(system, know(user, phone_number(N), hospital))*), we will have (using rule 38)

$$X = bel(user, by(call(user, hospital), talk(user, kathy), at(hospital, kathy)))$$

The system believes that the user's belief by calling to the hospital is to talk to Kathy (assuming she is at the hospital).

On the other hand, if the system already knows that Kathy is at home:

$$initiates(e, exp(system, at(home, kathy))), e < e1$$

then the belief that by calling to the hospital we will talk to Kathy is incorrect (from the system's point of view), and it is possible to detect that the user's expectation (at(hospital, kathy)) is different from the system's belief (at(home, kathy)). Taking these facts into account it is possible to plan the system's next actions.

9. CONCLUSIONS AND FUTURE WORK

We have 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. The framework allows to model a large class of cooperative information-seeking dialogues using a very general approach. In fact, the results obtained show that this framework is able to handle some of the classical problems that arise in dialogues (nonspecified goals, clarification subdialogues, error situations, misunderstandings).

This work has some connections with the work of D. Appelt and M. Pollack [2] using weighted abduction for plan ascription. In our work, we have the abductive reasoning integrated in a general nonmonotonic logic programming framework [13]. Moreover, we are using Event Calculus, and we have also connected the epistemic operators with a speech acts theory. In our approach, we describe the speech acts and the epistemic operators in terms of the A-language [7] and then we will apply a general transformation into abductive logic programs.

However, it is an open problem to prove the correctness of the abductive event calculus wrt the well founded semantics, and similarly, we have not proved the correctness of the transformations from the A-language into the event calculus logic programs.

Another open problem is related to the complexity of the system. In fact, the framework relies on algorithms that are very complex (for instance, finding the revision sets), and that complexity can cause computational problems if we are dealing with larger logic programs.

As future work, and in order to handle a larger class of problems, the framework should be incorporated in a more general natural language processing system such as the one described by Lopes and Quaresma [11, 15] where a multiheaded architecture coordinates several independent modules with the shared objective of supporting a robust natural language interaction.

We would like to thank the referees for their comments.

REFERENCES

1. J. F. Allen and C. R. Perrault. Analyzing intention in utterances. *Artificial Intelligence*, (15):143–178, 1980.
2. Douglas E. Appelt and Martha E. Pollack. Weighted abduction for plan ascription. *User Modeling and User-Adapted Interaction*, 2(1), 1992.
3. P. Cohen and H. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.
4. P. Cohen, J. Morgan, and M. Pollack. *Intentions in Communication*. MIT Press, Cambridge, MA, 1990.
5. Kave Eshghi. Abductive planning with event calculus. In *Proceedings of the International Conference on Logic Programming*, 1988.
6. 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.
7. M. Gelfond and V. Lifshitz. Representing actions in extended logic programs. In *Proceedings of the International Symposium on Logic Programming*, 1992.
8. J. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *Proceedings of the 26th Annual Meeting of ACL*, 1988.
9. D. Litman and J. Allen. A plan recognition model for subdialogues in conversations. *Cognitive Science*, (11):163–200, 1987.
10. Diane J. Litman. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues*. PhD thesis, Dep. of Computer Science, University of Rochester, 1985.
11. 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.
12. Lode Missiaen. *Localized Abductive Planning with the Event Calculus*. PhD thesis, Univ. Leuven, 1991.
13. 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.
14. Martha E. Pollack. *Inferring Domain Plans in Question-Answering*. PhD thesis, Dep. of Computer and Information Science, University of Pennsylvania, 1986.
15. 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.

16. P. Quaresma and J. G. Lopes. Abduction of plans and intentions in dialogues. In P. Codognet, P. M. Dung, and A. C. Kakas, editors, *Proceedings of the Workshop on Abductive Reasoning, International Conference on Logic Programming*, 1993.
17. P. Quaresma and J. G. Lopes. Reconhecimento de intenções para uma interacção robusta com bases de conhecimento médicas. In *EPLP'93 - Encontro Português da Língua Portuguesa*, 1993.
18. Earl D. Sacerdoti. *A Structure for Plans and Behavior.* American Elsevier, New York, 1977.
19. Murray P. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of the IJCAI*, 1989.