Modelling Agent Interaction in Logic Programming

Luís Moniz Pereira and Paulo Quaresma
{lmp|pq}@di.fct.unl.pt
CENTRIA - AI CENTRE
Departamento de Informática
Universidade Nova de Lisboa
2825 Monte da Caparica
Portugal

August 6, 1998

Abstract

We present a logic programming framework implemented over Prolog which is able to model an agent's mental state. An agent is modeled by a set of extended logic programming rules representing the agent's behavior, attitudes (beliefs, intentions, and goals), world knowledge, and temporal and reasoning procedures. At each stage the agents's mental state is defined by the well founded model of the extended logic program plus some constraints.

Via this modeling an agent is able to interact with other agents, updating and revising its mental state after each event. The revision process includes the ability to remove contradictions in the agent's mental state.

It is shown how this framework can handle interactions between agents with different behavior rules, namely, with different levels of cooperativeness and credulity.

1 Introduction

In order to interact with other agents, an agent needs the ability to model its mental state. Namely, it is necessary to represent its attitudes (beliefs, intentions, and goals), world knowledge and temporal, reasoning and behavior rules. We propose a logic programming framework that allows the representation of agent models and the definition of update and revise procedures to be executed after each event. This framework is based on a previous work of Quaresma [9] and it is implemented using the REVISE system [4] and the XSB-Prolog from SUNY at Stony Brook.

Agents are defined as logic programs extended with explicit negation and constraints and the semantics is given by the well founded semantics of logic programs with explicit negation (from Pereira et al. [1]). The well founded semantics has a complete and sound top-down proof procedure with polynomial complexity for finite programs.

At each new step, the agent's mental state is given by the well founded model of the logic program that models the agent after it has undergone possible revision. After each event, it is necessary to update the agent model with the new information. This is done through the updating of the logic program with the facts that describe the events. The update process may create a contradictory mental state. For instance, it is possible that an event initiates a belief that is contradictory with some previous beliefs. In these situations, it is necessary to revise the agent's mental state, terminating the attitudes that support the contradiction. This is achieved through the definition of revision preferences.

This approach also solves the existing problems in Giangrandi and Tasso work [6] where the revised model is obtained through the use of heuristics to obtain the minimal model. On the other hand, the update procedures allow us to reason about past and present attitudes and to build an intentional structure of the interaction. This an advantage over Ferguson's approach [5] where

his framework does not allow these kind of introspective reasoning. The use of extended logic programming with well founded semantics also avoids the complexity problems with the modal logic approach of Sadek [10] in his definition of rational agents.

In the next section, the logic programming framework is briefly described. In section 3 we present the agent modelling process, with a special focus on its capability to model agents with different levels of cooperativeness and credulity. The procedures to update and revise the agents' mental state after each event are described in section 4 and 5. Finally, in section 6 some conclusions and open problems are pointed out.

2 Logic programming framework

Logic programs extended with explicit negation are finite set of rules of the form

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

where H, B_1, ..., B_n, C_1,..., C_m are objective literals. An objective literal is an atom A or its explicit negation $\neg A$; *not* stands for negation by default; *not* L is a default literal. Literals are either objective or default and $\neg\neg L \equiv L$.

The set of all ground objective literals of a program P designates the extended Herbrand base of P and is represented by $\mathcal{H}(P)$. A partial interpretation I of an extended program P is represented by $T \cup not\ F$, where T and F are disjoint subsets of $\mathcal{H}(P)$. Objective literals in T are true in I; objective literals in F are false by default in I; objective literals of $\mathcal{H}(P) - (T \cup F)$ are undefined in I. Moreover, if $\neg L \in T$ then $L \in F$.

An interpretation I of an extended logic program P is a partial stable model of P iff $\Phi_P(I) = I$ (see [1] for the definition of the Φ operator).

The well founded model of the program P is the F-least partial stable model of P. The well founded semantics of P is determined by the set of all partial stable models of P.

Pereira et al. [1] showed that every non-contradictory program has a well founded model and they also presented a complete and sound top-down proof procedure for several classes of programs.

In their work, Pereira et al. proposed a revision process that restores consistency for contradictory programs, taking back assumptions of the truth value of negative literals. As will be described in section 4, we use a two-valued revision process where the selected model is the preferred extended stable model (XSM) obtained as the join of all minimal non-contradictory submodels (MNS). This model is called PCFXSM — Preferred Contradiction Free eXtended Stable Model (see [1, 9, 8]).

2.1 Events

In order to interact with other agents, an agent must be able to deal with time and events. In fact, it is very important that agents have the capability to represent time and events and they should be able to reason about their mental state at a given time point. They should also be able to change their mental state as a consequence of some external or internal events.

For a time formalism we use a variation of the Event Calculus [7] that allows events to have an identification and a duration and allows events to occur simultaneously. In this approach time is linear and discrete.

The predicate *holds_at* defining the properties that are true at a specific time is:

$$\begin{aligned} holds_at(P,T) \quad \leftarrow \quad & happens(E,T_i,T_f), \\ & initiates(E,T_P,P), \\ & T_P < T, \\ & persists(T_P,P,T). \end{aligned} \tag{1}$$

$$persists(T_P, P, T) \leftarrow not\ clipped(T_P, P, T). \quad (2)$$

$$clipped(T_P, P, T) \leftarrow happens(C, T_{ci}, T_{cf}), \quad (3)$$
$$terminates(C, T_C, P),$$
$$not\ out(T_C, T_P, T).$$

$$out(T_C, T_P, T) \leftarrow T \leq T_C. \quad (4)$$

$$out(T_C, T_P, T) \leftarrow T_C < T_P. \quad (5)$$

The predicate $happens(E, T_i, T_f)$ means that the event E took place, starting at T_i and ending at T_f; $initiates(E, T, P)$ means that the event E initiates fluent P at time T; $terminates(E, T, P)$ means that event E terminates P at time T; $persists(T_i, P, T)$ means that P persists from T_i until T (at least); $clipped(T_i, P, T)$ means that P was terminated in a time T_t that cannot be proved to be outside $[T_i, T[$; $out(T_t, T_i, T)$ means that $T \leq T_t$ (T is before the time that terminated the property P) or $T_t < T_i$ (the termination time is before the initiating time). There exists also the predicate $act(E, A)$ which states that action A is associated with event E; the predicate $ev_gen(P, T)$ which means that property P was generated before time T by an event; and the predicate $enabled(E, T_i)$ which means that the event E may occur at time T_i (its pre-conditions are satisfied);

Note that a property P is true at a time T ($holds_at(P, T)$), if there is a previous event that initiates P and if P persists until T. P persists until T if there cannot be proved by default the existence of another event that terminates P before time T.

We need additional rules for the relation between not holding a property and holding its negation and not holding the negation of a property and holding the property:

$$\neg holds_at(P, T) \leftarrow holds_at(\neg P, T). \quad (6)$$

$$\neg holds_at(\neg P, T) \leftarrow holds_at(P, T). \quad (7)$$

The above predicates require being related by some integrity rules in the form of denials[1]. Note that the constraints will be used in a 2-value revision. In this paper we will present only a general constraint relating a property and its negation (see [9] for a complete description):

$$\Leftarrow L, \neg L. \quad (8)$$

3 Agents' mental states

In our proposal, agents are modeled by the well founded model of an extended logic program with the following structure:

1. Rationality rules (RR). These rules describe the relation between the different attitudes (beliefs, intentions, and goals).
2. Behavior rules (BR). These rules define the agent's activity, cooperativeness, and credulity.
3. Actions description (Ac). These rules describe the actions that may be executed by the agent.
4. A temporal formalism (T). These are the rules presented in the previous section regarding events.
5. World knowledge (WK).These rules describe the agent's world knowledge: events, actions, entities, taxonomies.

In the next subsections we will analyze the first three structures: rationality rules, behavior rules, and actions description.

[1] We use the symbol $\Leftarrow$ to denote an integrity constraint.

3.1 Rationality rules

These rules define relations between the agents attitudes: beliefs (bel), goals ($goal$), and intentions (int). Beliefs are defined by the predicate $bel(A, P)$ which means that agent A beliefs in property P; goals are defined by the predicate $goal(A, P)$ which means that agent A wants property P to became true in the future; intentions are defined by the predicate $int(A, Act)$ which means that agent A has the intention to execute action Act.

Using this predicates it is possible to represent an agent's beliefs about other agents' beliefs or to represent introspective beliefs.

It is necessary to represent the relations and the constraints between these attitudes. In this paper we will present only two of these relations (for a complete description see [9, 3, 2]):

- Integrity

$$\Leftarrow \quad holds_at(bel(A, P), T),\ holds_at(bel(A, \neg P), T). \tag{9}$$

$$\Leftarrow \quad holds_at(goal(A, P), T),\ holds_at(goal(A, \neg P), T). \tag{10}$$

- Necessity

$$holds_at(bel(A, P), T) \ \leftarrow holds_at(P, T). \tag{11}$$

The necessity rule creates the problem of infinite loops, which can be solved in the implementation process by defining a maximum level of recursiveness [9].

3.2 Behavior rules

These rules allow the definition of the agent behavior. As behavior properties we have considered the credulity and cooperativeness.

3.2.1 Credulity

Credulity defines how an agent accepts new information.

The main process defines how beliefs are transferred:

$$\begin{aligned} holds_at(bel(H, P), T) \quad \leftarrow \quad & holds_at(bel(H, bel(S, P)), T), \\ & holds_at(transf(S, H, P), T). \end{aligned} \tag{12}$$

This rule defines that an agent believes in a proposition if he believes that another agent believes in it and if the information transfer process is valid.

The fundamental rule is the one that defines how an information transfer may occur. We have defined four types of transference from speaker to hearer:

1. The totally naïve agent that always accepts new information from a sincere speaker:

$$\begin{aligned} holds_at(transf(S, H, P), T) \quad \leftarrow \quad & holds_at(bel(H, naive(H)), T), \\ & holds_at(bel(H, sincere(S), T). \end{aligned}$$

   Note that this rule may create a contradictory state (initiating beliefs that are contradictory with previous ones). The contradiction removal process is described in section 5.

2. A credulous agent accepts new information if it doesn't contradict his previous beliefs:

$$\begin{aligned} holds_at(transf(S, H, P), T) \quad \leftarrow \quad & holds_at(bel(H, credulous(H)), T), \\ & not\ holds_at(bel(H, \neg P), T), \\ & holds_at(bel(H, sincere(S), T). \end{aligned} \tag{13}$$

3. A rational agent accepts the new information if it is plausible, i.e., if there exists an hypothetical sequence of actions that could achieve it:

$$\begin{aligned} holds_at(transf(S,H,P),T) \quad \leftarrow \quad & holds_at(bel(H,rational(H)),T), \\ & plausible(P), \\ & holds_at(bel(H,sincere(S),T). \end{aligned}$$

The *plausible* predicate is in fact a meta-predicate that creates an hypothetical model by adding a new integrity constraint to the existing model

$$IC' = IC \cup \{\Leftarrow not\ holds_at(P, t_\infty)\}$$

If it is possible to abduce a sequence of actions not limited in time that supports the property P (satisfying all integrity constraint), then the information P is plausible.

4. The skeptical agent never accepts new information from the other agents:

$$\begin{aligned} \neg holds_at(transf(S,H,P),T) \quad \leftarrow \quad & holds_at(bel(H,skeptical(H)),T), \\ & holds_at(bel(H,sincere(S),T). \end{aligned}$$

This kind of agent never learns from others, only from his own experiences.

3.2.2 Cooperativeness

This property defines how intentions and goals are transferred between agents.

For a totally cooperative agent:

$$\begin{aligned} holds_at(int(H,A),T) \quad \leftarrow \quad & holds_at(bel(H,int(S,A)),T), \\ & holds_at(bel(H,cooperative(H)),T). \end{aligned} \tag{14}$$

$$\begin{aligned} holds_at(goal(H,P),T) \quad \leftarrow \quad & holds_at(bel(H,goal(S,P)),T), \\ & holds_at(bel(H,cooperative(H)),T). \end{aligned} \tag{15}$$

Using these rules a cooperative agent accepts as new intentions and goals what he believes are the intentions and goals of his interlocutors.

The non-cooperative agents don't accept the intentions and goals of its interlocutors:

$$\begin{aligned} \neg holds_at(int(H,A),T) \quad \leftarrow \quad & holds_at(bel(H,int(S,A)),T), \\ & holds_at(bel(H,pass_non_coop(H)),T). \end{aligned} \tag{16}$$

$$\begin{aligned} \neg holds_at(goal(H,P),T) \quad \leftarrow \quad & holds_at(bel(H,goal(S,P)),T), \\ & holds_at(bel(H,pass_non_coop(H)),T). \end{aligned} \tag{17}$$

3.3 Actions description

The agents' model must have a description of the actions that may be executed by agents.

Actions are described in terms of their pre-conditions and effects. For instance, if action A has pre-conditions $P_1, ..., P_n$ and it has the effect F, then it can be represented by the following set of rules:

$$\begin{aligned} enabled(E, T_i) \quad &\leftarrow \quad act(E, A), \\ &\qquad holds_at(P_1, T_i), \\ &\qquad ..., \\ &\qquad holds_at(P_n, T_i). \\ initiates(E, T_f, F) \quad &\leftarrow \quad act(E, A), \\ &\qquad happens(E, T_i, T_f), \\ &\qquad holds_at(P_1, T_i), \\ &\qquad ...., \\ &\qquad holds_at(P_n, T_i). \end{aligned}$$

Note that in the *initiates* predicate we need to test explicitly the action's pre-conditions because we may have different effects depending on the pre-conditions. For instance, to turn on the car key may cause different effects depending on whether the car has fuel or not.

4 Updating an agent's mental state

The agent's mental state, as defined in the previous sections, must be updated after each event.

This process is defined through the use of logic program updates in the following way:

Definition 1 *Let P be the agent logic program at a given time: $P = RR \cup Ac \cup T \cup BR \cup WK$, where RR are the rationality rules, Ac are the rules defining the domain actions, T are the temporal axioms presented in section 2.1, BR are the behavior rules, and WK are the world knowledge rules (including the events representation).*

Let E be the logic program representing events $e_1, ..., e_n$ with its associated actions $a_1, ..., a_m$.

E :

$$\begin{aligned} &act(e_1, a_1). \\ &happens(e_1, t_1, t_1'). \ ...happens(e_n, t_n, t_n'). \\ &act(e_{1,} a_1). \ ...act(e_{n,} a_m). \end{aligned}$$

The new agent's attitudes At are the properties $bel/2$, $goal/2$, and $int/2$) that hold in the Preferred Contradiction Free eXtended Stable Model (PCFXSM) of the updated program $P \cup E$:

$$holds_at(At, t) \in PCFXSM(P \cup E)$$

The update process may initiate some attitudes which are inconsistent with the previous mental state.

As an example, suppose agent a believes at time t_1, that Kathy is at the hospital:

$$holds_at(bel(a, at(hospital, kathy)), t_1).$$

At a greater time point, $t_2 > t_1$, he is informed that she is at home:

$$\begin{aligned} &happens(e_1, t_2, t_2). \\ &act(e_1, inform(b, a, at(home, kathy))). \end{aligned}$$

If agent a is naïve, he will adopt the new information. However, there might exist an integrity constraint stating that is contradictory to believe that an agent may be at two different places at the same time:

$$\begin{aligned} \Leftarrow \quad & holds_at(bel(A, at(B, L_1)), T), \\ & holds_at(bel(A, at(B, L_2)), T), \\ & L_1 \neq L_2. \end{aligned}$$

In this situation, the model must be revised, and some attitudes should be terminated. This process is handled through the calculation of the PCFXSM and will be described in the next section.

5 Revising Mental States

Contradiction may be caused by the effects of the new events (these effects may violate some integrity constraints).

In fact, contradictions caused by the effects of events can be associated with integrity constraint rules of the following form:

$$\Leftarrow holds_at(P_1, T), holds_at(P_2, T).$$

Suppose that an event initiated property P_2, and property P_1 is also valid; a possible approach could be to terminate P_1 (or P_2). This can be done adding the property $terminates(E, T, P_1)$ to the set of revisables and defining an order relation between the contradiction free extended stable models.

Given a set of minimal non-contradictory submodels, it is necessary to define a procedure that creates an order between them and chooses the preferred solution. Depending on the application domain, it might be better to have a conservative approach (keeping the oldest attitudes) or a more progressive one (keeping the new attitudes). The order relation is built using the time when each property was initiated: $initiates(E, T, P)$. The sum of these times gives an order between the models (mapping each model to its summing value).

So, for each model m at a given time T we obtain a value v:

$$\begin{aligned} v = & \sum_{holds_at(P,T) \in m} T_P, \text{where } T_P \text{ is such that} \\ & initiates(E, T_P, P) \in m, \text{ and} \\ & \forall T, T < T_P, initiates(E, T, P) \notin m \end{aligned}$$

Using this process if we want a conservative behavior we should choose the model with the minimum value; if we want a more progressive behavior we should choose the model with maximum value.

As an example, suppose the situation of the previous section where agent a believes Kathy is at the hospital and he is informed that she is at home:

$$\begin{aligned} & holds_at(bel(a, em(hospital, kathy)), t_2). \\ & holds_at(bel(a, em(home, kathy)), t_2). \end{aligned}$$

We have the integrity constraint, shown before:

$$\begin{aligned} \Leftarrow \quad & holds_at(bel(X, at(L_1, Y)), T), \\ & holds_at(bel(X, at(L_2, Y), T), \\ & L_1 \neq L_2. \end{aligned}$$

The revision process obtains the non-contradictory solutions and it also obtains the preferred solution (accordingly with a pre-defined order criteria):

1. Terminate the belief that Kathy is at the hospital;
2. Terminate the belief that Kathy is at home.

6 Conclusions

We have proposed an agent modeling process with the following characteristics:

1. It was defined over a logic programming framework implemented over Prolog;
2. It allows the definition of reasoning and behavior rules. These rules allow the modeling of non-well behaved agents;
3. It has an update and revise procedure defined for any event that may occur.

Moreover this framework integrates under a logic programming environment temporal reasoning, user modeling, update programs, and contradiction removal mechanisms.

As future work we also intend to integrate this agent modeling framework in a more general architecture allowing a more powerful representation of interactions. Namely, the architecture should be modular and distributed and it should be able to deal with interruptions and real time problems.

Acknowledgements

This work was partially supported by PRAXIS project MENTAL (2/2.1/TIT/1593/95), and DIXIT (2/2.1/TIT/1670/95).

References

[1] José Júlio Alferes and Luís Moniz Pereira. *Reasoning with Logic Programming*, volume 1111 of *Lecture Notes in Artificial Intelligence*. Springer, 1996.

[2] Michael Bratman. *What is Intention?, in Intentions in Communication*. MIT, 1990.

[3] Philip Cohen and Hector Levesque. Communicative actions for artificial agents. In *ICMAS*, pages 65–72, 1995.

[4] Carlos Damásio, Wolfgang Nejdl, and Luís Moniz Pereira. Revise: An extended logic progamming system for revising knowledge bases. In Morgan Kaufmann, editor, *KR'94*, 1994.

[5] George Ferguson. *Knowledge Representation and Reasoning for Mixed-Initiative Planning*. PhD thesis, University of Rochester, 1995.

[6] Paolo Giangrandi and Carlo Tasso. Managing temporal knowledge in student modeling. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *Proceedings of the 6th International Conference on User Modeling*, pages 414–426. SpringerWienNewYork, 1997.

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

[8] Luís Moniz Pereira, José Júlio Alferes, and Joaquim Nunes Aparício. The extended stable models of contradiction removal semantics. In P. Barahona, L. M. Pereira, and A. Porto, editors, *5th Portuguese AI Conference, Volume 541 of LNAI*, pages 105–119. Springer-Verlag, 1991.

[9] Paulo Quaresma. *Inferência de Atitudes em Diálogos*. PhD thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa, 1997. In Portuguese.

[10] M. Sadek, P. Bretier, and F. Panaget. Artimis: Natural dialogue meets rational agency. In Martha Pollack, editor, *IJCAI'97 - 15th International Conference on Artificial Intelligence*, pages 1030–1035. Morgan Kaufmann, 1997.