Expressing Preferences Declaratively in Logic-based Agent Languages

Stefania Costantini and **Arianna Tocchio**
Università degli Studi di L'Aquila
Dipartimento di Informatica
Via Vetoio, Loc. Coppito,
I-67010 L'Aquila - Italy
Email: {stefcost, tocchio } @di.univaq.it

Pierangelo Dell'Acqua
Department of Science and
Technology - ITN
Linköping University
601 74 Norrköping, Sweden
Email: pier@itn.liu.se

Abstract

Preference-based reasoning is a form of commonsense reasoning that makes many problems easier to express and sometimes more likely to have a solution. In this paper, we present an approach to introducing preferences among actions in logic-based agent-oriented languages. These preferences are local to the rule where they are defined. To the best of our knowledge, no similar approach has been proposed before, and cannot be easily simulated by means of preferences expressed in a global way. The approach is exemplified by choosing preferred actions to be performed in reaction to an event.

Introduction

Expressing and using preferences constitutes a form of commonsense reasoning for several reasons. Intuitively, it simulates a skill that every person takes for granted. From the point of view of knowledge representation, many problems are more naturally represented by flexible rather than by hard descriptions. Practically, many problems would not even be solvable if one would stick firmly on all requirements. Consider for instance the well-known situation of persons who would not like to be seated at the same table with somebody they hate, and prefer to be seated together with somebody they like: preferring instead of absolutely expecting make more instances of the problem solvable.

Intelligent agents perform advanced activities such as negotiation, bargaining, etc. where they have to choose among alternatives. The choice may be supported by some kind of preference or priorities related for instance to the agent's objectives, the context (cooperative vs. competitive), available resources, strategies that the agent intends to follow. Preferences are in fact an instance of those *approximate concepts* that John McCarthy advocates in (McCarthy 1996) as essential on the way of machines reaching human-level AI.

Agents will in general include specialized modules and/or meta-level axioms for applying priorities and preferences: see for instance (Gelfond & Son 1998) for a seminal work on prioritized defeasible reasoning. However, it can be useful for logical agents to be able to express preferences at a more basic linguistic level. These basic preferences can then be employed in building more advanced high-level strategies.

At the language level, various forms of preferences can be expressed in Answer Set Programming (ASP), which is a recent and now well-established paradigm of logic programming (Gelfond & Lifschitz 1988; Lifschitz 1999). ASP is not based on answering queries, but rather it is based on computing sets of consequences of a given program ("Answer Sets") that fulfill the underlying semantics. In ASP, the basic mechanism for managing preferences is that of computing the Answer Sets and then choose the "preferred" ones. Below we will review relevant existing approaches on preference reasoning. Some of them are based on establishing priorities/preferences among atoms (facts), and typically introduce some form of disjunction in the head (conclusion) of rules. In contrast, other approaches express priorities among rules.

Our proposal is aimed at allowing agents defined in any rule-based computational logic language to express preferences concerning either which action they would perform in a given situation, or, in perspective, which goal they would pursue at a certain stage. Since actions are often performed in reaction to events and goals are set in accordance to some internal conclusion that has been reached, we propose to introduce disjunction in the body (conditions) of reactive rules (that specify how to cope with events). If the body of a reactive rule contains a disjunction among two or more actions, the *preferred* one will be chosen according to *preference condition expressions*, as (following (Sakama & Inoue 2000)) priorities may be conditional. These expressions are local to the rule where the disjunction occurs, so that we achieve a better *elaboration tolerance* (McCarthy 1998) by allowing different preferences among the same actions in different contexts.

If agents evolve with time and enlarge their knowledge and experience, choices performed according to priorities may dynamically change according to the agent evolution. In fact, preference condition expressions may contain references to the agent past experience, and then the preferred choice may be different at different stages.

In agent languages that are not based on Answer Set Programming, one cannot select the preferred model(s) by means of a filter on possible models. Then, other techniques are needed in order to provide a semantic account. Recently, a general semantic framework for logical rule-based agent-oriented languages has appeared in the literature (Costantini & Tocchio 2006): in this framework, an agent is understood as an entity coping with *events*. Events of any kind are considered as making a change in the agent program, which is a logical theory, and in its semantics (however defined). For such a change to be represented, the effect of an event is understood as the application of a corresponding program-transformation function. Thus, agent evolution is seen as program evolution, with a corresponding semantic evolution.

This semantic framework can be applied to the approach presented here by adapting the proposal of the *split programs* introduced in (Brewka 2002). A split program is a version of the given program obtained by replacing each disjunction by one of its options. Then, at each step where an agent must prefer one option out of a disjunction of possibilities, we would have a set of *possible evolutions*, each corresponding to a split program. Among them, the preferred one (according to the present conditions) is taken, while all the others are pruned.

In the rest of the paper, we first briefly review previous related work on preferences and discuss how it relates to ours. Then, we introduce the approach at more length, and discuss its semantics. Finally, we close with some remarks and proposals of future work.

Previous Related Work

The reader may refer to (Delgrande *et al.* 2004) for a discussion of many existing approaches to preferences. The main distinction is among those that define priorities/preferences among atoms (facts), and typically introduce some form of disjunction in the head of rules, and those that express instead priorities among rules. Among the latter ones, we mention (Kakas & Moraitis 2006) that applies preferences among rules in negotiating agents based on argumentation, so as to tune argumentations according to changing contexts.

The approach of (Sakama & Inoue 2000) considers *general extended disjunctive programs* where a rule has the syntax:

$$L_1| \dots |L_k| not\, L_{k+1}| \dots |not\, L_{k+h} \leftarrow Body$$

where "|" represents disjunction and not is negation as failure under the Answer Set semantics. A preference, or priority, between two ground literals e_1, e_2 is expressed in the form $e_1 \prec e_2$. An answer set S_2 of a given program is preferable onto another answer set S_1 iff $S_2 \setminus S_1$ contains an element e_2 whose priority is higher than some element e_1 in $S_1 \setminus S_2$, and the latter does not contain another element e_3 whose priority is strictly higher that e_2. Then, preferred answer sets (or p-answer sets) are a subset of the traditional ones, that can be seen as a special case corresponding to empty priorities.

Basic PLP is exploited in (Sakama & Inoue 2000) so as to express priorities not only between facts, but also between more general forms of knowledge. The approach allows many forms of commonsense reasoning to be modeled.

An interesting application is that of *priority with preconditions*. For instance, borrowing the example from (Sakama & Inoue 2000), the situation where a person drinks tea or coffee but she prefers coffee to tea when sleepy can be represented as follows (in a prolog-like syntax):

$$\begin{array}{l} tea \mid coffee. \\ tea \prec coffee \text{ :- } sleepy. \end{array}$$

This program can be translated in a standard way in plain PLP and, assuming that *sleepy* holds, has the p-answer set $\{sleepy, coffee\}$.

In LPODS (Brewka 2002), one can write expressions such as $A \times B$ in the head of rules, where the new connective $\times$ stands for ordered disjunction. The expression intuitively stands for: if possible A, but if A is impossible then (at least) B. If there are several disjuncts, the first one represents the best preferred option, the second one represents the second best option, etc. The following is an example where a person who wishes to spend the evening out and has money prefers to go to theatre, or else (if impossible) to go to the cinema, or else (if both previous options cannot be taken) to go to dine at a restaurant.

$$\begin{array}{l} theatre \times cinema \times restaurant \text{ :- } \\ \qquad\qquad want\_to\_go\_out, have\_money. \end{array}$$

For selecting the preferred answer set(s) of a program P, one obtains the possible split programs of P, where a split program P' is obtained from P by replacing each disjunctive rule by one of its options. Then, the answer sets of P are taken to be the answer sets of the split programs. To choose preferred ones given that there may be several disjunctions, a notion of *degree of satisfaction* of disjunctive rules must be defined, that induces a partial ordering on answer sets. Preferred answer sets are those that satisfy all rules of P to the better degree.

In the approach that we propose here for logical agents, a disjunction among two or more actions may occur in the body of a reactive rule that specifies the response to an event. The intended meaning is the following:

- preference condition expressions, which are local to the rule where the disjunction occurs, establish which action is preferred under which conditions;
- precondition of actions state (globally) in which cases an action can be actually performed, i.e., when it is feasible;
- having to choose, the agent should perform the best preferred feasible action.

To the best of our knowledge, the approach that we propose here of introducing conditional preferences in the body of logical rules is novel, and no similar other approach can be found in the literature. Our approach cannot be easily simulated by using preferences in the head. Consider in fact that preferences defined in the head are *global*, while instead, preferences expressed in the body are *local* to the rule where they occur. Thus, there may be different preferences among the same actions in different contexts, which implies a better *elaboration tolerance* (McCarthy 1998) as local preferences related to managing a certain event can be modified without affecting other aspects of the agent behavior. This will be demonstrated in the following examples.

The application of preference reasoning to address the problem of action selection in multi-agent systems is also new. As an agent evolves in time and its knowledge changes, preferred choices will change as well. Then, according to the same preference structure an agent will in general prefer differently at different stages of its life. Of course we build upon previous work, in particular we adopt the idea of conditional preferences from (Sakama & Inoue 2000) and for the semantics we use the idea of split programs introduced in (Brewka 2002).

The approach in more detail

The constructs that we introduce below can be easily employed in any agent-oriented languages based on logic (horn-clause) programming, and can be adapted to rule-based languages in general. Similarly to (Sakama & Inoue 2000), we assume the following:

- preferences are expressed between two ground facts;
- preferences are expressed explicitly by means of special rules that may have conditions;
- the preference relation is transitive, irreflexive and anti-symmetric.

Preferences can be defined conditionally among actions that agents may perform. We make some preliminary assumption about the agent languages we are considering. We do not commit to any particular syntax, though we will propose a sample one in order to introduce and illustrate examples. We will discuss the semantics of the class of languages that we consider in the next section. By saying "an agent" we mean a program written in the language at hand, that behaves as an agent when it is put at work. We assume in particular the following syntactic and operational features.

- The agent is able to perceive external events coming from the environment where the agent is situated. In our sample syntax an external event is an atom which is distinguished by a postfix E. E.g., $rainE$ indicates an external event.
- The agent is able to react to external events, i.e., the language provides some kind of condition-action construct. In our sample syntax a reactive rule will be indicated with $pE :> Body$ meaning that whenever the external event pE is perceived, the agent will execute $Body$. There are languages (like, e.g., the one presented in (Costantini & Tocchio 2002)) where an agent can react to its own internal conclusions, that are interpreted as events (thus modeling proactivity). We assume that the syntax for reaction is the same in both cases. However, an internally generated event is indicated with postifix I, i.e., in the form pI.
- The agent is able to perform actions. Actions will occur in the agent program as special atoms. In our sample syntax we assume them to be in the form qA, i.e., they are distinguished by suffix A. E.g., $open\_umbrellaA$ indicates an action. Actions may have preconditions: In our sample syntax we assume them to be expressed by rules. The connective $:<$ indicates that the rule defines the precondition of an action. I.e., a precondition rule will be indicated as $qA :< Body$, meaning that the action qA can be performed only if $Body$ is true. We do not cope here with the effective execution of actions, that is left to the language run-time support.

The proposed approach will be presented by adopting and extending our sample syntax. In particular, a disjunction (indicated with "|") of actions may occur in the body of a reactive rule. Then, a rule $pE :> q1A \,|\, q2A, Body$. means that in reaction to pE the agent may perform indifferently either action $q1A$ or action $q2A$ and then it executes $Body$. Preferences among actions are defined in *preference condition expressions* associated to a reactive rule, and indicated by the new connective $<<$. Then, a rule $pE :> q1A \,|\, q2A, Body \;::\; q1A << q2A :\!\!- Conds$. means that in reaction to pE the agent may perform either action $q1A$ or action $q2A$, but action $q2A$ is preferred over action $q1A$ provided that $Conds$ holds. I.e., if $Conds$ is not verified then the preference is not applicable, and thus any of the actions can be indifferently executed. In general, a disjunction may contain several actions, and several preference condition expression can be expressed, provided that they refer to distinct pairs of actions (for every two action, at most one preference condition expression can be specified).

A set of preference rules define in general a *partial order* among actions, where preferences are transitively applied and actions that are unordered can be indifferently executed. In our approach preferences are applied on *feasible* actions. I.e., the partial order among actions must be re-evaluated at each step of the agent life where a choice is possible, according to the preconditions of the actions. The preferred actions at each stage are those that can actually be performed and that are selected by the preference partial order.

Example 1 *Consider a person who receives an invitation to go out for dinner. She would prefer accepting the invitation rather than refusing, provided that the invitation comes from nice people. She is able to accept if she has time. The invitation is an* external event *that reaches the agent from her external environment. Accepting or refusing constitutes the* reaction *to the event, and both are actions. One of the actions (namely, accepting) has preconditions. In our sample syntax, an agent program fragment formalizing this situation may look as follows.*

$invitation\_dinnerE :> acceptA \,|\, refuseA ::$
$\qquad refuseA << acceptA :\!\!- nice\_people\_inviting.$
$acceptA :< have\_time.$

When the external event $invitation\_dinnerE$ *is perceived by the agent, it can react by alternatively performing one of two actions. The action* $acceptA$ *will be performed only* *if its preconditions are verified. As preferences are among feasible actions, $acceptA$ is preferred provided that both $have\_time$ and $nice\_people\_inviting$ hold.*

Notice that while preference condition expressions are local to a (reactive) rule, action preconditions are global and must be verified whenever an action is attempted.

Notice also that what the agent will do is not known in advance, as the agent evolves in time: the invitation may arrive at a stage of the agent operation when time is available, and then the preferred action is chosen. If instead the invitation (or, another future invitation) arrives when there are no resources for accepting, then the agent will have to refuse.

Consider instead a person that receives an invitation to a boring work meeting. She will prefer (unconditionally) to refuse. However, she cannot do that if she does not have an excuse to present. As we can see, preference among the same actions varies according to the context. Also, if the preconditions of a preferred action are not verified, a less preferred one will have to be performed.

$$
\begin{aligned}
&invitation\_meetingE :> acceptA \mid refuseA ::\\
&\qquad acceptA << refuseA.\\
&refuseA :< acceptable\_excuse.
\end{aligned}
$$

Another example will introduce further aspects.

Example 2 *Let us now rephrase the example of the person preferring coffee over tea if sleepy. Let us put it in a proactive perspective, where the person wonders whether it is time to take a break from working, e.g., at mid-afternoon. If so, she will consider whether to drink tea or coffee or juice. Moreover, in this variation the agent drinks coffee only if she can have an espresso and drinks juice only if she can have orange juice.*

The corresponding program fragment might look as follows, where $take\_break$ is an internal conclusion *that triggers a proactive behavior: the first rule reaches the conclusion that taking a break is in order; the second rule states what to do then, i.e., specifies a reaction to the internal conclusion itself (indicated in the second rule with postfix I for "internal"). For the mechanism to be effective, $take\_break$ must be attempted from time to time, so as to trigger the consequent behavior as soon as it becomes true.*

$$
\begin{aligned}
&take\_break :\text{-}\ five\_oclock.\\
&take\_breakI :> drink\_teaA \mid drink\_coffeeA\\
&\qquad\qquad \mid drink\_juiceA ::\\
&\qquad drink\_teaA << drink\_coffeeA :\text{-}\ sleepy,\\
&\qquad drink\_teaA << drink\_juiceA.\\
&drink\_coffeeA :< espresso.\\
&drink\_juiceA :< orange.
\end{aligned}
$$

The expected behavior is the following:

- If $sleepy$ holds and $espresso$ holds as well, the agent can drink coffee (the action $drink\_coffeeA$ is allowed) and will not drink tea, which is less preferred. If $orange$ holds, also the action $drink\_juiceA$ is allowed, and preferred over $drink\_teaA$. The agent can indifferently drink either coffee or juice, as they are unrelated. If $orange$ does not hold, the agent will definitely drink coffee.
- If $espresso$ does not hold, the agent cannot drink coffee (the action $drink\_coffeeA$ is not allowed). Then, if $orange$ holds then the agent will drink juice (the action $drink\_juiceA$ will be performed), otherwise it will drink tea (as the action $drink\_teaA$ is always allowed, not having preconditions).
- If $sleepy$ does not hold, there is no preference between tea and coffee. If $orange$ does not hold and $espresso$ holds, one of the two actions $drink\_teaA$ or $drink\_coffeeA$ can be indifferently executed. If $orange$ holds and $espresso$ holds as well, $drink\_juiceA$ is preferred over $drink\_teaA$, but as no other priority is specified, one of the actions $drink\_coffeeA$ or $drink\_juiceA$ can be indifferently executed.

Declarative Semantics of Evolving Agents with Preferences

The evolutionary semantics proposed in (Costantini & Tocchio 2006) has the objective of providing a unifying framework for various languages and semantics of reactive, proactive logic-based agents. This semantic approach is based upon declaratively modelling the changes inside an agent which are determined by changes in the environment as well as agent's own self-modifications. The key idea is to understand these changes as the result of the application of program-transformation functions. In this view, a program-transformation function is applied upon reception of an event, internal or external to the agent. In fact, the perception of an event affects the program defining the agent: for instance, an event can be stored as a new fact in the program. Similarly, actions which are performed can be recorded as new facts. All the "past" events and actions will constitute the "experience" of the agent.

Recording each event or action or any other change that occurs inside an agent can be semantically interpreted as transforming the agent program into a new program, that may procedurally behave differently than before: e.g., by possibly reacting to the event, or drawing conclusions from past experience. Furthermore, the internal event corresponding to the decision of the agent to undertake an activity triggers a more complex program transformation, resulting in version of the program where the corresponding *intention* is somewhat "loaded" so as to become executable.

Then, every agent will be equipped with an initial program P_0 which, according to these program-transformation steps (each one transforming P_i into P_{i+1}), gives rise to a Program Evolution Sequence $PE = [P_0, ..., P_n]$. The program evolution sequence will have a corresponding Semantic Evolution Sequence $ME = [M_0, ..., M_n]$ where M_i is the semantic account of P_i according to the specific language and the chosen semantics. The couple $\langle PE; ME\rangle$ is called the *Evolutionary Semantics* of the agent program P_{Ag}, corresponding to the particular sequence of changes that has happened. The evolutionary semantics represents the history of an agent without having to introduce the concept of "state".

Various agent languages and formalisms will influence the following key points:

1. When a transition from P_i to P_{i+1} takes place, i.e., which are the external and/or internal factors that determine a change in the agent.
2. Which kind of transformations are performed.
3. Which semantic approach is adopted, i.e., how M_i is obtained from P_i. M_i can be for instance a model or an initial algebra. In general, given a semantics $\mathcal{S}$ we will have $M_i = \mathcal{S}(P_i)$.

A transition from P_i to P_{i+1} can reasonably take place, for instance: when an event happens; when an action is performed; when a new goal is set; upon reception of new knowledge from other agents; in consequence to the decision to accept/reject the new knowledge; in consequence to the agent decision to revise its own knowledge. We say that at stage P_{i+1} of the evolution the agent *has perceived* event ev meaning that the transition from P_i to P_{i+1} has taken place in consequence of the reception of ev. It is reasonable to assume that at the stage P_{i+1} the agent will cope with the event by executing a selected reaction.

In our approach we perform an *Initialization step* by which the program P_{Ag}, written by the programmer, is transformed into a corresponding initial program P_0 via some sort of knowledge compilation. This compilation can on one extreme do nothing, while on the other extreme it can perform complex transformations by producing "code" that implements language features in the underlying logical formalism. P_0 can be simply a program (logical theory) or can have additional information associated to it.

To give a semantic account to a program we proceed as follows: (i) in the initialization step preferences can be collected and preference conditions removed; (ii) P_0 will not contain preference conditions, but will be associated to a *preference structure Pref* where, for each reactive rule occurring in P_{Ag}, preferences between couples of (ground) actions are made explicit by aggregating preference condition expressions and annotating conditions[1]. Then, $P_{Ag} \rightarrow^{Initialization\_Step} \langle P_0, Pref \rangle$

To give a semantic account to reactive rules according to preferences, we adapt the idea of *split program* from (Brewka 2002). A split program is a version of the given program obtained by replacing a disjunction by one of its disjuncts. In our case, whenever an agent at stage P_i of its evolution has perceived an (either external or internal) event, say pE, it will react to it. However, if there is a disjunction of actions in the body of the corresponding reactive rule, then the agent may react in more that one way. The different ways of reacting are represented by different *split programs*, each one representing an alternative.

[1]This is by no means a trivial task if preferences are partially ordered. For possible solutions the reader may refer for instance to (Dell'Acqua & Pereira 2005; Junker & Kiessling 2006; Pini *et al.* 2006)

Definition 1 *Let P_{Ag} be an agent program that has been transformed into a program P_0 (associated with a preference structure $Pref$) by the initialization step. Let P_i be the program obtained from the evolution of P_0 at the i-th step, corresponding to the perception of event pE. Let $pE :> Body$ be the corresponding reactive rule in P_i, where a disjunction of actions occurs in Body. A* split program P_i' *is obtained by replacing the disjunction with one of its options.*

Referring to the program of Example 1, at the initialization step it is transformed into:

$$invitation\_dinnerE :> acceptA \mid rejectA.$$
$$acceptA :< have\_time.$$

where the preference $refuseA << acceptA$:- $nice\_people\_inviting$ is recorded in the structure $Pref$, in association to the reactive rule. Then, the perception of $invitation\_dinnerE$ will give two split programs: the first one ϕ_1 where the body of the reactive rule contains only $acceptA$ and the second ϕ_2 where the body of the reactive rule contains $refuseA$. We will have a set $\{P_i^1, \ldots, P_i^k\}$ of split programs corresponding to the number k of actions occurring in the disjunction. Assuming that events are considered one at a time (i.e., an evolution step copes with a single event), at each stage split programs will be relative to a single reactive rule, and will correspond to a set $\{M_i^1, \ldots, M_i^k\}$ where M_i^j is the semantics of P_i^j. We say that *a split occurs* at stage P_i of the program evolution whenever at that stage the incoming event is related to a reactive rule with a disjunction of actions in its body. The preferred split programs are those whose semantics contain the preferred actions.

Definition 2 *Let P_{Ag} be an agent program that has been transformed into a program P_0 by the initialization step, and let Pref be the preference structure that has been associated to the program. Let P_i correspond to a step of the evolution of P_0, where a split occurs. Given two split programs P_i^r and P_i^s obtained from P_i by splitting a disjunction $d \equiv act^1A \mid \ldots \mid act^kA$, then P_i^r is preferred over P_i^s if the following conditions hold:*

- *the semantics M_i^r of P_i^r contains act^rA_i;*
- *act^rA_i is preferred over act^sA_i according to Pref, which implies that M_i^r entails the conditions of the related preference condition expressions.*

Notice that both M_i^r and M_i^s may not contain the corresponding action (act^rA_i and act^sA_i respectively), in case its preconditions are false. Then, a split program is preferred upon another one if (i) its semantics entails the related action and (ii) either the semantics of the other one does not entail the related action, or the former action is preferred and the conditions for preference satisfied.

At each step where a split occurs we have a set of *possible evolutions*, each corresponding to a split program. Among them the preferred one (according to the present conditions) is taken, while all the others are pruned. As mentioned before, given similar situations at different stages of the agent life, different options can be taken, according to the present assessment of the agent knowledge. In the previous example, ϕ_1 will be preferred to ϕ_2 whenever it actually entails $acceptA$.

We can have a unique most preferred split program P_i^b if $Pref$ is a total order with respect to the actions over which we split, or we may have more than one equally preferred split programs. Any of them can be indifferently selected.

Definition 3 *Let P_i be a stage of the program evolution sequence where a split occurs. We let P_{i+1} be any of the most preferred split programs.*

Concluding Remarks

In this paper we have presented an approach to expressing preferences among actions and in logical agents. The approach builds on previous relevant work related to answer set programming, but is rephrased for reactive and proactive agents that evolve in time. Other approaches in computational logic that are related to the present one and to which we are indebted are (Alferes, Dell'Acqua, & Pereira 2002; Dell'Acqua & Pereira 2003;), where preferences and updating preferences are coped with in the context of a more general approach to updating logic programs. The examples that we have presented basically refer to the DALI language (Costantini & Tocchio 2002; 2004), in which the present proposal is being implemented and will be experimented.

Our next research aim is to extend the possibility of expressing preferences to all kinds of subgoals occurring in the body of logical rules, instead of coping with actions only. Another important objective is to extend the approach so as to be able to express preferences among agent goals (objectives to reach). Actually in fact, a plan for reaching an objective can be seen as divided into: (i) a preliminary check stage, where feasibility of subsequent actions is checked (are the tickets available? Do I have the money? Do my friends accept to join me? May I rent a car?); (ii) an operative stage, where actions that influence the environment (and in general cannot be retracted, or at least not so easily) are performed. The first stage can be seen as a feasibility stage for setting an objective. Then, if there is a disjunction of objectives in the body of a rule, we intend to make an agent able to choose the most preferred feasible one.

References

Alferes, J. J.; Dell'Acqua, P.; and Pereira, L. M. 2002. A compilation of updates plus preferences. In *Logics in Artificial Intelligence, Proc. of the 8th Europ. Conf., JELIA 2002*, LNAI 2424, 62–74. Springer-Verlag, Berlin.

Brewka, G. 2002. Logic programming with ordered disjunction. In *Proc. of AAAI-02, Edmonton, Canada*.

Costantini, S., and Tocchio, A. 2002. A logic programming language for multi-agent systems. In *Logics in Artificial Intelligence, Proc. of the 8th Europ. Conf., JELIA 2002*, LNAI 2424. Springer-Verlag, Berlin.

Costantini, S., and Tocchio, A. 2004. The dali logic programming agent-oriented language. In *Logics in Artificial Intelligence, Proc. of the 9th European Conference, Jelia 2004*, LNAI 3229. Springer-Verlag, Berlin.

Costantini, S., and Tocchio, A. 2006. About declarative semantics of logic-based agent languages. In Baldoni, M., and Torroni, P., eds., *Declarative Agent Languages and Technologies*, LNAI 3229. Springer-Verlag, Berlin. Post-Proc. of DALT 2005.

Delgrande, J.; Schaub, T.; Tompits, H.; and Wang, K. 2004. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence*.

Dell'Acqua, P., and Pereira, L. M. Preferential Theory Revision. J. of Applied Logics. Special issue: Formal and Computational Epistemology. To appear in 2006.

Dell'Acqua, P., and Pereira, L. M. 2003. Preferring and updating in logic-based agents. In *Web-Knowledge Management and Decision Support*, LNAI 2543. Springer-Verlag, Berlin. 70–85. Selected Papers from the 14th Int. Conf. on Applications of Prolog (INAP).

Dell'Acqua, P., and Pereira, L. M. 2005. Preference Revision via Declarative Debugging. In Bento, C.; Cardoso, A.; and Dias, G., eds., *Progress in Artificial Intelligence, Procs. 12th Portuguese Intl.Conf. on Artificial Intelligence (EPIA'05)*, LNAI 3808, 29–42. Springer-Verlag.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of 5th ILPS conference*, 1070–1080.

Gelfond, M., and Son, T. C. 1998. Reasoning with prioritized defaults. In *Proc. of the 3rd Int. Works. on Logic Programming and Knowledge Representation*, LNAI 1471, 164–223. Springer-Verlag, Berlin.

Junker, U., and Kiessling, W., eds. 2006. *Proc. of multidisciplinary ECAI06 Workshop about Advances on Preference Handling*.

Kakas, A., and Moraitis, P. 2006. Adaptive agent negotiation via argumentation. In *Proc. of the 5th International Joint Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'06)*.

Lifschitz, V. 1999. Answer set planning. In *Proc. of ICLP '99 Conf.*, 23–37. The MIT Press. Invited talk.

McCarthy, J. 1996. From here to human-level ai. In *Proc. of the Fifth Int. Conf. on Principles of Knowledge Represent. and Reasoning (KR'96)*. Invited talk.

McCarthy, J. 1998. Elaboration tolerance. In *Proc. of Common Sense'98*. Available at http://www-formal.stanford.edu/jmc/ elaboration.html.

Pini, M. S.; Rossi, F.; Venable, K.; and Walsh, T. 2006. Incompleteness and incomparability in preference aggregation. In *Proc. of multidisciplinary ECAI06 Workshop about Advances on Preference Handling*.

Sakama, C., and Inoue, K. 2000. Prioritized logic programming and its application to commonsense reasoning. *Artif. Int.* 123(1-2):185–222.