Hierarchical Decision Making By Autonomous Agents

Stijn Heymans, Davy Van Nieuwenborgh[⋆], and Dirk Vermeir[⋆⋆]

Dept. of Computer Science
Vrije Universiteit Brussel, VUB
Pleinlaan 2, B1050 Brussels, Belgium
{sheymans,dvnieuwe,dvermeir}@vub.ac.be

Abstract. Often, decision making involves autonomous agents that are structured in a complex hierarchy, representing e.g. authority. Typically the agents share the same body of knowledge, but each may have its own, possibly conflicting, preferences on the available information.
We model the common knowledge base for such preference agents as a logic program under the extended answer set semantics, thus allowing for the defeat of rules to resolve conflicts. An agent can express its preferences on certain aspects of this information using a partial order relation on either literals or rules. Placing such agents in a hierarchy according to their position in the decision making process results in a system where agents cooperate to find solutions that are jointly preferred.
We show that a hierarchy of agents with either preferences on rules or on literals can be transformed into an equivalent system with just one type of preferences. Regarding the expressiveness, the formalism essentially covers the polynomial hierarchy. E.g. the membership problem for a hierarchy of depth n is Σ^P_{n+2}-complete. We illustrate an application of the approach by showing how it can easily express a generalization of weak constraints, i.e. "desirable" constraints that do not need to be satisfied but where one tries to minimize their violation.

1 Introduction

In *answer set programming*[16, 2] one uses a logic program to modularly describe the requirements that must be fulfilled by the solutions to a particular problem, i.e. the answer sets of the program must correspond to the intended solutions of the problem. The technique has been successfully applied to the area of agents and multi-agent systems[3, 8, 26]. While [3] and [8] use the basic answer set semantics to represent the agents domain knowledge, [26] applies an extension of the semantics incorporating preferences among choices in a program.

The idea of extending answer set semantics with some kind of preference relation is not new. We can identify two directions for these preferences relations on programs. On the one hand, we can equip a logic program with a preference relation on the rules [18,

[⋆] Supported by the FWO

[⋆⋆] This work was partially funded by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-37004 WASP project.

$17, 15, 10, 7, 5, 27, 1, 22$], while on the other hand we can consider a preference relation on the (extended) literals in the program: [21] proposes explicit preferences while [4, 6] encodes dynamic preferences within the program.

The traditional answer set semantics is not universal, i.e. programs may not have any answer sets at all. This behavior is not always feasible, e.g. a route planner agent may contain inconsistent information regarding some particular regions in Europe, which should not stop it from providing travel directions in general. The extended answer set semantics from [22, 23] allows for the *defeat* of problematic rules. Take, for example, the program consisting of $a \leftarrow b$, $b \leftarrow$ and $\neg a \leftarrow$. Clearly this program has no answer sets. It has, however, extended answer sets $\{a, b\}$, where the rule $\neg a \leftarrow$ is defeated by the applied $a \leftarrow b$, and $\{\neg a, b\}$, where $a \leftarrow b$ is defeated by $\neg a \leftarrow$.

However, not all extended answer sets may be equally preferred by the involved parties: users traveling in "error free" regions of Europe do not mind faults in answers concerning the problematic regions, in contrast to users traveling in these latter regions that want to get a "best" approximation. Therefore, we extend the above semantics by equipping programs with a preference relation over either the rules or the literals in a program. Such a preference relation can be used to induce a partial order on the extended answers, the minimal elements of which will be preferred.

Different agents may exhibit different, possibly contradicting, preferences, that need to be reconciled into commonly accepted answer sets, while taking into account the relative authority of each agent.

For example, sending elderly workers on early pension, reducing the wages, or sacking people are some of the measures that an ailing company may consider. On the other hand, management may be asked to sacrifice expense accounts and/or company cars. Demanding efforts from the workers without touching the management leads to a bad image for the company. Negotiations between three parties are planned: shareholders, management and unions. The measures under consideration, together with the influence on the company's image are represented by the extended answer sets

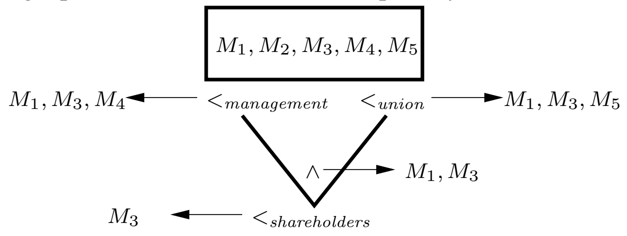
$$\begin{array}{ll} M_1 = \{bad\_image, pension\} & M_4 = \{\neg bad\_image, expense, sack\} \\ M_2 = \{bad\_image, wages\} & M_5 = \{\neg bad\_image, car, wages\} \\ M_3 = \{\neg bad\_image, expense, wages\} \ . & \end{array}$$

The union representative, who is not allowed to reduce the options of the management, has a preference for the pension option over the wages reduction over the sacking option of people, not taking into account the final image of the company, i.e. $pension < wages < sack < \{bad\_image, \neg bad\_image\}$. This preference strategy will result in M_1 being better than M_2, while M_3 is preferred upon M_4. Furthermore, M_5 is incomparable w.r.t. the other options. Thus M_1, M_3 and M_5 are the choices to be defended by the union representative. Management, on the other hand, would rather give up its expense account than its car, regardless of company image, i.e. $expense < car < \{bad\_image, \neg bad\_image\}$, yielding M_1, M_3 and M_4 as negotiable decisions for the management.

Finally, the shareholders take only into account the decisions that are acceptable to both the management and the unions, i.e. M_1 and M_3, on which they apply their own preference $\neg bad\_image < bad\_image$, i.e. they do not want their company to get a bad image. As a result, $M_3 \sqsubset M_1$, yielding that M_3 is the preferred way to go to save the company, taking into account each party's preferences.

Decision processes like the one above are supported by *agent hierarchies*, where a program, representing the shared world of agents, is equipped with a tree of preference relations on either rules or literals, representing the hierarchy of agents preferences. Semantically, preferred extended answer sets for such systems will result from first optimizing w.r.t. the lowest agents in the hierarchy, then grouping the results according to the hierarchy and let the agents on the next level optimize these results, etc. Thus, each agent applies its preferences on a selection of the preferred answers of the agents immediately below it in the hierarchy, where the lowest agents apply their preferences directly on the extended answer sets of the shared program.

Reconsidering our example results in the system depicted below, i.e. union and management prefer directly, and independently, among all possible solutions, while the shareholders only choose among the solutions preferred by both union and management, obtaining a preferred solution for the complete system.



Such agent hierarchies turn out to be rather expressive. More specifically, we show that such systems can solve arbitrary complete problems of the polynomial hierarchy. We also demonstrate how systems with combined preferences, i.e. either on literals or on rules, can effectively be reduced to systems with only one kind of preference.

Finally, we introduce a generalization of weak constraints[9], which are constraints that should be satisfied but may be violated if there are no other options, i.e. violations of weak constraints should be minimized. Weak constraints have useful applications in areas like planning, abduction and optimizations from graph theory[13, 11]. We allow for a hierarchy of agents having their individual preferences on the weak constraints they wish to satisfy in favor of others. We show that the original semantics of [9] can be captured by a single preference agent.

The remainder of the paper is organized as follows. In Section 2, we present the extended answer set semantics together with the hierarchy of preference agents, enabling hierarchical decision making. The complexity of the proposed semantics is discussed in Section 3. Before concluding and giving directions for further research in Section 5, we present in Section 4 a generalization of weak constraints and show how the original semantics can be implemented. Due to lack of space, detailed proofs have been omitted.

2 Agent Hierarchies

We give some preliminaries concerning the extended answer set semantics[22]. A *literal* is an atom a or a negated atom $\neg a$. An *extended literal* is a literal or a literal preceded by the *negation as failure*-symbol *not*. A program is a countable set of rules of the form

$\alpha \leftarrow \beta$ with α a set of literals, $|\alpha| \leq 1$, and β a set of extended literals. If $\alpha = \emptyset$, we call the rule a *constraint*. The set α is the *head* of the rule while β is called the *body*. We will often denote rules either as $a \leftarrow \beta$ or, in the case of constraints, as $\leftarrow \beta$. For a set X of literals, we take $\neg X = \{\neg l \mid l \in X\}$ where $\neg\neg a$ is a; X is *consistent* if $X \cap \neg X = \emptyset$. The positive part of the body is $\beta^+ = \{l \mid l \in \beta, l \text{ literal}\}$, the negative part is $\beta^- = \{l \mid \textit{not } l \in \beta\}$, e.g. for $\beta = \{a, \textit{not } \neg b, \textit{not } c\}$, we have that $\beta^+ = \{a\}$ and $\beta^- = \{\neg b, c\}$. The *Herbrand Base* $\mathcal{B}_P$ of a program P is the set of all atoms that can be formed using the language of P. Let $\mathcal{L}_P$ be the set of literals and $\mathcal{L}_P^*$ the set of extended literals that can be formed with P, i.e. $\mathcal{L}_P = \mathcal{B}_P \cup \neg\mathcal{B}_P$ and $\mathcal{L}_P^* = \mathcal{L}_P \cup \{\textit{not } l \mid l \in \mathcal{L}_P\}$. An *interpretation* I of P is any consistent subset of $\mathcal{L}_P$. For a literal l, we write $I \models l$, if $l \in I$, which extends for extended literals *not* l to $I \models \textit{not } l$ if $I \not\models l$. In general, for a set of extended literals X, $I \models X$ if $I \models x$ for every extended literal $x \in X$. A rule $r : a \leftarrow \beta$ is *satisfied* w.r.t. I, denoted $I \models r$, if $I \models a$ whenever $I \models \beta$, i.e. r is *applied* whenever it is *applicable*. A constraint $\leftarrow \beta$ is satisfied w.r.t. I if $I \not\models \beta$. The set of satisfied rules in P w.r.t. I is the *reduct* P_I. For a *simple* program P (i.e. a program without *not*), an interpretation I is a *model* of P if I satisfies every rule in P, i.e. $P_I = P$; it is an *answer set* of P if it is a minimal model of P, i.e. there is no model J of P such that $J \subset I$. For programs P containing *not*, we define the *GL-reduct* w.r.t. an interpretation I as P^I, where P^I contains $\alpha \leftarrow \beta^+$ for $\alpha \leftarrow \beta$ in P and $\beta^- \cap I = \emptyset$. I is an *answer set* of P if I is an answer set of P^I. A rule $a \leftarrow \beta$ is *defeated* w.r.t. I if there is a *competing* rule $\neg a \leftarrow \gamma$ that is applied w.r.t. I, i.e. $\{\neg a\} \cup \gamma \subseteq I$. An *extended answer set* I of a program P is an answer set of P_I such that all rules in $P \backslash P_I$ are *defeated*.

Example 1. Take a program P expressing an intention to vote for either the Democrats or the Greens. Voting for the Greens will, however, weaken the Democrats, possibly resulting in a Republican victory. Furthermore, you have a Republican friend who may benefit from a Republican victory.

$$\begin{array}{rlrl}
dem\_vote & \leftarrow & \neg dem\_vote & \leftarrow \\
green\_vote & \leftarrow \textit{not } dem\_vote & rep\_win & \leftarrow green\_vote \\
fr\_benefit & \leftarrow rep\_win & \neg fr\_benefit & \leftarrow rep\_win
\end{array}$$

This program results in 3 different extended answer sets $M_1 = \{dem\_vote\}$, $M_2 = \{\neg dem\_vote, green\_vote, rep\_win, fr\_benefit\}$, and $M_3 = \{\neg dem\_vote, green\_vote, rep\_win, \neg fr\_benefit\}$.

As mentioned in the introduction, the background knowledge for *agents* will be described by a program P. Agents can express individual preferences either on extended literals or on rules of P, corresponding to *literal* and *rule* agents respectively.

Definition 1. *Let P be a program. A* ***rule agent*** *(RA) $\mathcal{A}$ for P is a well-founded strict partial*[1] *order $<$ on rules in P. The order $<$ induces a relation $\sqsubseteq$ among interpretations M and N of P, such that $M \sqsubseteq N$ iff $\forall r_2 \in P_N \backslash P_M \cdot \exists r_1 \in P_M \backslash P_N \cdot r_1 < r_2$.*

[1] A strict partial order on X is an anti-reflexive and transitive relation on X. A strict partial order on a finite X is *well-founded*, i.e. every subset of X has a minimal element w.r.t. $<$.

*A **literal agent** (LA) for P is a strict well-founded partial order $<$ on $\mathcal{L}^*_P$, and $M \sqsubseteq N$ iff $\forall n \in \{l \in \mathcal{L}^*_P \mid N \models l \wedge M \not\models l\}, \exists m \in \{l \in \mathcal{L}^*_P \mid M \models l \wedge N \not\models l\} \cdot m < n$.*

*The extended answer sets of an agent for P correspond to the extended answer sets for P. As usual, we have $M \sqsubset N$ iff $M \sqsubseteq N$ and not $N \sqsubseteq M$. A **preferred answer set** M is an extended answer set that is minimal w.r.t. $\sqsubset$ among the extended answer sets.*

Note that a RA $<$ for P corresponds to an *ordered logic program* (OLP) $\langle P, < \rangle$ from [22].

We refer to the order of an agent $\mathcal{A}$ with $<_{\mathcal{A}}$ and $\sqsubset_{\mathcal{A}}$. Intuitively, for rule agents, an extended answer set M is "better" than N if each rule that is satisfied by N but not by M is countered by a better rule satisfied by M and not by N. Similarly, for literal agents we have that $M \sqsubseteq N$ if every extended literal that is true in N, but not in M, is countered by a better one true in M but not in N.

E.g., define a rule agent $\mathit{fr\_benefit} \leftarrow \mathit{rep\_win} < \neg \mathit{fr\_benefit} \leftarrow \mathit{rep\_win}$ for the program P in Example 1, indicating that one rather satisfies the former rule than the latter. We have, with $P_{M_1} = P \backslash \{\neg \mathit{dem\_vote} \leftarrow\}$, $P_{M_2} = P \backslash \{\mathit{demo\_vote} \leftarrow, \neg \mathit{fr\_benefit} \leftarrow \mathit{rep\_win}\}$ and $P_{M_3} = P \backslash \{\mathit{demo\_vote} \leftarrow, \mathit{fr\_benefit} \leftarrow \mathit{rep\_win}\}$, that $M_2 \sqsubset M_3$, yielding that M_1 and M_2 are the only preferred answer sets.

A literal agent might insist on voting for the Democrats: $\mathit{demo\_vote} < \mathcal{L}^*_P \backslash \{\mathit{demo\_vote}\}$, making M_1 its only preferred answer set.

The cooperation of agents for a program P is established by arranging them in a tree-structure[2], such that decisions are made bottom-up, starting with agents that have no successors, all the way up in the hierarchy to the root agent, each agent processing the results of its successor agents. Formally, an *agent hierarchy* (AH) is a pair $\langle P, T \rangle$ where P is a program and T is a finite and/or-tree of agents $\mathcal{A}$ for P.

We will denote the root agent $\mathcal{A}$ of the tree T with $\mathcal{A}_\varepsilon$.The m successors of an agent $\mathcal{A}_x$ are denoted as $\mathcal{A}_{x \cdot 1}, \ldots, \mathcal{A}_{x \cdot m}$. An agent without successors is called an *independent agent*, other agents are *dependent*. An agent associated with an and-node (or-node) will be called an and-agent (or-agent). We define what it means for an extended answer set to be *preferable* by a certain agent in the hierarchy.

Definition 2. *Let $\langle P, T \rangle$ be an AH. An extended answer set M of P is **preferable** by an independent agent $\mathcal{A}$ of T if M is a preferred answer set of $\mathcal{A}$ for P. An extended answer set M of P is preferable by a dependent and-agent (or-agent) $\mathcal{A}_x$, with m successors, if*

- *M is preferable by every (some) $\mathcal{A}_{x \cdot i}$, $1 \leq i \leq m$, and*
- *there is no N, preferable by every (some) $\mathcal{A}_{x \cdot j}$, $1 \leq j \leq m$, such that $N \sqsubset_{\mathcal{A}_x} M$.*

*An extended answer set M of P is **preferred** if it is preferable by $\mathcal{A}_\varepsilon$.*

Rule agents (OLPs) are rather convenient to formulate diagnostic problems[24, 25], using "normal" and "fault" model rules to describe the system under consideration, where the former are preferred over the latter. Examples of this approach can be found in [24, 25], where it also has been shown that the OLP semantics yields minimal possible explanations. However, to decide which explanations to check first, an engineer

[2] For simplicity we restrict ourselves to trees, however, the results remain valid for any well-founded strict partial order of agents that has a unique maximal agent.

typically uses another preference order, preferring e.g. explanations that are cheaper to verify. Such a situation can be modelled by an agent hierarchy containing an extra agent "above" the diagnostic RA. More generally, one may imagine situations where multiple engineers each have their own experience (expressed by preference) and where the head of the group has to take the final decision on which possible explanations to check first, taking into account the proposals of her colleagues. Such systems can easily be expressed using the proposed framework.

Reasoning w.r.t. agent hierarchies, containing both rule and literal agents, can be reduced to reasoning w.r.t. rule agent hierarchies (RAHs) or literal agent hierarchies (LAHs), i.e. hierarchies containing only rule or literal agents.

We show the reduction from RAs to LAs and vice versa. For the reduction of RAs for P to LAs, we introduce for every rule r in P a corresponding atom that is in an answer set iff r is satisfied. Intuitively, the newly introduced atoms will be ordered according to the original order on the rules they correspond with.

Theorem 1. *Let P be a program and $R = \{r_i \leftarrow \mathit{not}\, b \mid r_i : \alpha \leftarrow \beta \in P, b \in \beta^+\} \cup \{r_i \leftarrow b \mid r_i : \alpha \leftarrow \beta \in P, b \in \beta^-\} \cup \{r_i \leftarrow a \mid r_i : a \leftarrow \beta \in P\}$ with a new atom r_i for each rule r_i in P. M is a preferred answer set of a RA $\mathcal{A}^r$ for P iff $M' = M \cup \{r_i \mid r_i \in P_M\}$ is a preferred answer set of the LA $\mathcal{A}^l$ for $P \cup R$ where $\{r_i\} <_{\mathcal{A}^l} \mathcal{L}^*_P \cup \mathit{not}(\{r_i\})$ with additionally $r_i <_{\mathcal{A}^l} r_j$ iff $r_i <_{\mathcal{A}^r} r_j$.*

Moreover, preferred answer sets of a LA for P are in one to one correspondence with the preferred answer sets of a LA for $P \cup R$ by simply ignoring the newly added atoms r_i.

Theorem 2. *Let P be a program and R as in Theorem 1. M is a preferred answer set of a LA $\mathcal{A}$ for P iff $M' = M \cup \{r_i \mid r_i \in P_M\}$ is a preferred answer set of the LA $\mathcal{A}'$ for $P \cup R$ where $<_{\mathcal{A}'}$ is equal to $<_{\mathcal{A}}$ with additionally $k <_{\mathcal{A}'} \mathcal{L}^*_{P \cup R} \backslash \mathcal{L}^*_P$ for every extended literal k appearing in $<_{\mathcal{A}}$.*

The opposite simulation of a LA by a RA can be done by introducing for each literal l and its extended version *not* l rules $l' \leftarrow$ and $\neg l' \leftarrow$ and ordering those rules according to the order on the extended literals.

Theorem 3. *Let P be a program and $L = \{l' \leftarrow\, ; \neg l' \leftarrow \mid l \in \mathcal{L}_P\} \cup \{\leftarrow l', \mathit{not}\, l;\ \leftarrow \neg l', l \mid l \in \mathcal{L}_P\}$. M is a preferred answer set of a LA $\mathcal{A}^l$ for P iff $M' = M \cup \{(\neg)l' \mid l \in \mathcal{L}_P, M \models (\mathit{not})l\}$ is a preferred answer set of the RA $\mathcal{A}^r$ for $P \cup L$ with $L <_{\mathcal{A}^r} P$ and additionally $(\neg)l' \leftarrow\ <_{\mathcal{A}^r} (\neg)k' \leftarrow$ iff $(\mathit{not})l <_{\mathcal{A}^l} (\mathit{not})k$.*

Example 2. Take a LA $\mathcal{A}^l$ for P where P consists of the rules $b \leftarrow a$, $a \leftarrow$, and $\neg a \leftarrow$, and $\neg a <_{\mathcal{A}^l} \{a, b, \mathit{not}\, \neg a\}$. This agent has two extended answer sets $\{\neg a\}$ and $\{a, b\}$, of which the first one is preferred. The corresponding RA $\mathcal{A}^r$ is defined by the following program[3]

$b \leftarrow a$	$a \leftarrow$	$\neg a \leftarrow$	
$a' \leftarrow$	$b' \leftarrow$	$(\neg a)' \leftarrow$	$(\neg b)' \leftarrow$
$\neg a' \leftarrow$	$\neg b' \leftarrow$	$\neg(\neg a)' \leftarrow$	$\neg(\neg b)' \leftarrow$
$\leftarrow a', \mathit{not}\ a$	$\leftarrow b', \mathit{not}\ b$	$\leftarrow (\neg a)', \mathit{not}\ \neg a$	$\leftarrow (\neg b)', \mathit{not}\ \neg b$
$\leftarrow \neg a', a$	$\leftarrow \neg b', b$	$\leftarrow \neg(\neg a)', \neg a$	$\leftarrow \neg(\neg b)', \neg b$

[3] Rules below the line are smaller than the ones above w.r.t. $<_{\mathcal{A}^r}$.

and $(\neg a)' \leftarrow\ <_{\mathcal{A}^r} \{a' \leftarrow\ , b' \leftarrow\ , \neg(\neg a)' \leftarrow\ \}$. This RA has the preferred answer set $\{\neg a, (\neg a)', \neg a', \neg b', \neg(\neg b)'\} = \{\neg a\} \cup \{(\neg)l' \mid \{\neg a\} \models (\mathit{not})l\}$.

Similarly to Theorem 2, we have that RAs for P can be simulated by RAs for $P \cup L$.

Theorem 4. *Let P be a program and L as in Theorem 3. M is a preferred answer set of a RA $\mathcal{A}$ for P iff $M' = M \cup \{(\neg)l' \mid l \in \mathcal{L}_P, M \models (\mathit{not})l\}$ is a preferred answer set of a RA $\mathcal{A}'$ for $P \cup L$ where $<_{\mathcal{A}'}$ is equal to $<_{\mathcal{A}}$ with additionally $r <_{\mathcal{A}'} L$ for every r in P.*

Theorem 1 and 2 allow the simulation of an arbitrary AH by a LAH. This is done by extending the program P with the set of rules R as in Theorem 1, and by transforming the rule agents to literal agents (Theorem 1), while the literal agents are adapted according to Theorem 2.

Theorem 5. *Let $\langle P, T\rangle$ be an AH. M is a preferred answer set of $\langle P, T\rangle$ iff $M' = M \cup \{r_i \mid r_i \in P_M\}$ is a preferred answer set of the LAH $\langle P \cup R, T'\rangle$, with T' defined as T but with every rule or literal agent replaced by a literal agent as in Theorems 1 and 2.*

Similarly, but now with Theorems 3 and 4, we can reduce arbitrary AHs to RAHs.

Theorem 6. *Let $\langle P, T\rangle$ be an AH. M is a preferred answer set of $\langle P, T\rangle$ iff $M' = M \cup \{(\neg)l' \mid l \in \mathcal{L}_P, M \models (\mathit{not})l\}$ is a preferred answer set of the RAH $\langle P \cup L, T'\rangle$, with T' defined as T but with every rule or literal agent replaced by a rule agent as in Theorems 3 and 4.*

3 Complexity

We briefly recall some relevant notions of complexity theory (see e.g. [20, 2] for a nice introduction). The class P (NP) represents the problems that are deterministically (nondeterministically) decidable in polynomial time, while $coNP$ contains the problems whose complement are in NP.

The polynomial hierarchy, denoted PH, is made up of three classes of problems, i.e. Δ_k^P, Σ_k^P and Π_k^P, $k \geq 0$, which are defined as $\Delta_0^P = \Sigma_0^P = \Pi_0^P = P$, $\Delta_{k+1}^P = P^{\Sigma_k^P}$, $\Sigma_{k+1}^P = NP^{\Sigma_k^P}$, and $\Pi_{k+1}^P = co\Sigma_{k+1}^P$. The class $P^{\Sigma_k^P}$ ($NP^{\Sigma_k^P}$) represents the problems decidable in deterministic (nondeterministic) polynomial time using an oracle for problems in Σ_k^P, where an oracle is a subroutine capable of solving Σ_k^P problems in unit time. The class PH is defined by $PH = \bigcup_{k=0}^{\infty} \Sigma_k^P$. Note that $\Sigma_k^P \subseteq \Sigma_k^P \cup \Pi_k^P \subseteq \Delta_{k+1}^P \subseteq \Sigma_{k+1}^P$. In the following, we will usually omit the P-superscript to avoid cluttered up lines. A language L is called complete for a complexity class C if both L is in C and L is hard for C. Showing that L is hard is normally done by reducing a known complete decision problem to a decision problem in L.

First of all, checking whether an interpretation I is an extended answer set of a program P is in P, because (a) checking if each rule in P is either satisfied or defeated w.r.t. I, (b) applying the GL-reduct on P_I w.r.t. I, i.e. computing $(P_I)^I$, and (c) checking whether the positive program $(P_I)^I$ has I as its unique minimal model, can all be done in polynomial time.

For an agent $\mathcal{A}$ and a program P, checking whether M is not a preferred answer set is in NP, because one can guess a set $N \sqsubset_{\mathcal{A}} M$ in polynomial time, and subsequently verify that N is an extended answer set of P, which can also be done in P.

On the other hand, the complexity of checking whether an extended answer set M is not preferable by a certain agent $\mathcal{A}_x$ in a hierarchy $\langle P, T \rangle$ depends on the location of the agent in the tree T. For an agent $\mathcal{A}_x$ in T, we denote with $d(\mathcal{A}_x)$ the length of the longest path from $\mathcal{A}_x$ to an independent agent $\mathcal{A}_{x \cdot y}$, $y \in \mathbb{N}^\star$, i.e. $d(\mathcal{A}_x) = \max_{\mathcal{A}_{x \cdot y}} |y|$ over independent agents $\mathcal{A}_{x \cdot y}$, where $|y|$ is the length of the string y. We define the depth of T as the longest path from the root, i.e. $d(T) = d(\mathcal{A}_\varepsilon)$.

Lemma 1. *Let $\langle P, T \rangle$ be an AH*[4]*, and let M be an extended answer set of P. Checking whether M is not preferable by $\mathcal{A}_x$ is in $\Sigma_{d(\mathcal{A}_x)+1}$.*

Proof. The proof is by induction. In the base case, i.e. $\mathcal{A}_x$ is an independent agent, we have that $d(\mathcal{A}_x) = 0$. Checking whether M is not preferable by $\mathcal{A}_x$ means checking whether M is not a preferred answer set of the agent $\mathcal{A}_x$ for P, which is in $NP = \Sigma_1 = \Sigma_{d(\mathcal{A}_x)+1}$.

For the induction step, checking that M is not preferable by a dependent and-agent (or-agent) $\mathcal{A}_x$ with m successors can be done by (a) checking that M is (or is not) preferable by every (some) $\mathcal{A}_{x \cdot i}$, $1 \leq i \leq m$. Since checking whether M is (or is not) preferable by an $\mathcal{A}_{x \cdot i}$ can be done, by the induction hypothesis, in $\Sigma_{d(\mathcal{A}_{x \cdot i})+1}$, we have that checking whether M is preferable by an $\mathcal{A}_{x \cdot i}$ is also in $C \equiv \Sigma_{\max_{1 \leq i \leq m} d(\mathcal{A}_{x \cdot i})+1}$, and (b) guessing, if M is preferable by every (some) $\mathcal{A}_{x \cdot i}$, $1 \leq i \leq m$, an interpretation $N \sqsubset_{\mathcal{A}_x} M$ and checking that it is not the case that N is not preferable by every (some) $\mathcal{A}_{x \cdot i}$, $1 \leq i \leq m$, which is again in C due to the induction hypothesis.

As a result, at most $2m$ calls are made to a C-oracle and at most one guess is made, yielding that the problem itself is in $NP^C = \Sigma_{\max_{1 \leq i \leq m} d(\mathcal{A}_{x \cdot i})+1+1} = \Sigma_{d(\mathcal{A}_x)+1}$. □

Using the above yields the following theorem about the complexity of AHs.

Theorem 7. *Let $\langle P, T \rangle$ be an AH and l a literal. Deciding whether there is a preferred answer set containing l is in $\Sigma_{d(T)+2}$. Deciding whether every preferred answer set contains l is in $\Pi_{d(T)+2}$.*

Proof. The first task can be performed by an NP-algorithm that guesses an interpretation $M \ni l$ and checks that it is not the case that M is not preferable up to the root agent $\mathcal{A}_\varepsilon$. Due to Lemma 1, the latter is in $\Sigma_{d(\mathcal{A}_\varepsilon)+1} = \Sigma_{d(T)+1}$, so the former is in $NP^{\Sigma_{d(T)+1}} = \Sigma_{d(T)+2}$.

By the previous, finding a preferred answer set M not containing l, i.e. $l \notin M$, is in $\Sigma_{d(T)+2}$. Hence, the complement of the problem is in $\Pi_{d(T)+2}$. □

Consider a LAH $\langle P, T \rangle$ where the tree T is a linear order containing n literal agents $\{\mathcal{A}_\varepsilon, \mathcal{A}_1, \mathcal{A}_{11}, \ldots \mathcal{A}_{11 \ldots 1}\}$, i.e. *a linear LAH*. Deciding whether there is a preferred answer set of a linear LAH, containing a literal, is Σ_{n+1}-complete, i.e. $\Sigma_{d(T)+2}$-complete, as is shown in [19]. Furthermore, deciding whether every preferred answer set of a linear LAH contains a literal, is $\Pi_{d(T)+2}$-complete [19]. Hardness for AHs follows then immediately from the hardness of linear LAHs.

[4] The depth of the tree is assumed to be bounded by a constant.

Theorem 8. *The problem of deciding, given an AH $\langle P, T \rangle$ and a literal l, whether there exists a preferred answer set containing l is $\Sigma_{d(T)+2}$-hard. Deciding whether every preferred answer set contains l is $\Pi_{d(T)+2}$-hard.*

Proof. Checking whether there is a preferred answer set containing l for a linear LAH $\langle P, T \rangle$ is $\Sigma_{d(T)+2}$-complete, and since a linear LAH is a AH, the result follows.

The second problem can be similarly shown to be $\Pi_{d(T)+2}$-hard. □

The following is immediate from Theorem 7 and 8.

Corollary 1. *The problem of deciding, given an arbitrary AH $\langle P, T \rangle$ and a literal l, whether there is a preferred answer set containing l is $\Sigma_{d(T)+2}$-complete. On the other hand, deciding whether every preferred answer set contains l is $\Pi_{d(T)+2}$-complete.*

4 Relationship with Weak Constraints

Weak constraints were introduced in [9] as a relaxation of the concept of a constraint. Intuitively, a weak constraint is allowed to be violated, but only as a last resort, meaning that one tries to minimize the number of violated constraints. Additionally, weak constraints may be hierarchically layered by means of a totally ordered set of sets of weak constraints $W = \{W_1, W_2, \ldots, W_n\}$, where it is assumed that $W_i < W_{i+1}$, $1 \leq i < n$, if the weak constraints in W_i are more important than the ones in W_{i+1}. Intuitively, one first chooses the answer sets that minimize the number of violated constraints in the most important W_1, and then, among those, one chooses the extended answer sets that minimize the number of violated constraints in W_2, etc.

Formally, a *weak logic program* (WLP) is a pair $\langle P, W \rangle$ where P is a program, and W is a totally ordered set of sets of weak constraints, specified syntactically as constraints $\leftarrow \beta$.

Definition 3. *Let $\langle P, W = \{W_1, \ldots, W_n\} \rangle$ be a WLP. An extended answer set M of P is* ***preferable*** *up to W_1 if no extended answer set N of P exists such that $|V^N_{W_1}| < |V^M_{W_1}|$, where $V^X_{W_i}$ are the weak constraints in W_i that are violated by an interpretation X. An extended answer set M of P is preferable up to W_i, $1 < i \leq n$, if*

- *M is preferable up to W_{i-1}, and*
- *there is no N, preferable up to W_{i-1}, such that $|V^N_{W_i}| < |V^M_{W_i}|$.*

An extended answer set M of P is ***preferred*** *if it is preferable up to W_n.*

In [9] a *Datalog*^{not}-program LP is used[5] disallowing empty heads and classical negation, but allowing for a set of *strong* constraints S. Clearly, this is subsumed by Definition 3, by taking $P = LP \cup S$, and noting that the extended answer set semantics reduces to the answer set semantics, due to the absence of classical negation. Although the preferred models are defined in [9] by means of an object function that has to be minimized, they are equivalent[12] to the ones resulting from Definition 3.

[5] The general mechanism is introduced with *Datalog*$^{\vee,not}$-programs, which allow for disjunction in the head.

The semantics of weak constraints, with preferability up to certain levels, appears very similar to our preferability notion in an agent hierarchy. However, due to the use of cardinality, deciding whether a literal l is contained in some preferred answer set of a WLP is Δ_2^P-complete. As $\Delta_2^P \subseteq \Sigma_2^P$, agent hierarchies of depth 0 suffice to capture WLPs. More specifically, we show that a single agent can solve the problem.

Example 3. Take the weak logic program $\langle P, \{W_1, W_2\}\rangle$, with the program P consisting of rules $a \leftarrow$, $\neg a \leftarrow$, $b \leftarrow$, and $\neg b \leftarrow$, and $W_1 = \{\leftarrow a\}$, $W_2 = \{\leftarrow \neg a, \leftarrow \neg b\}$. We have 4 extended answer sets $M_1 = \{a, b\}$, $M_2 = \{a, \neg b\}$, $M_3 = \{\neg a, b\}$, and $M_4 = \{\neg a, \neg b\}$ of which M_3 and M_4 are preferable up to W_1, and only M_3 is preferable up to W_2. Indeed $|V_{W_1}^{M_1}| = |V_{W_1}^{M_2}| = 1$, $|V_{W_1}^{M_3}| = |V_{W_1}^{M_4}| = 0$, $|V_{W_2}^{M_1}| = 0$, $|V_{W_2}^{M_2}| = |V_{W_2}^{M_3}| = 1$, and $|V_{W_2}^{M_4}| = 2$. We define the set WC as the rules $c_1^1 \leftarrow a$, $c_1^2 \leftarrow \neg a$, and $c_2^2 \leftarrow \neg b$, identifying the weak constraints and the level on which they appear, and rules counting the number of violated constraints in a W_i, for $0 \leq l \leq k$,

$$\begin{array}{ll} co(1,0,w_i) \leftarrow not\ c_1^i & co(2,l,w_2) \leftarrow co(1,l,w_2), not\ c_2^2 \\ co(1,1,w_i) \leftarrow c_1^i & co(2,l+1,w_2) \leftarrow co(1,l,w_2), c_2^2 \end{array}$$

Intuitively, the third argument in a *co/3* literal identifies the particular W_i we are looking at, the first argument shows the number of constraints in W_i that have already been considered, and the second argument effectively counts the number of violated constraints in W_i. Further, WC also contains the rules defining the number of violated constraints in each set of weak constraints, $i \in \{0, 1\}$, $j \in \{0, 1, 2\}$: $co(i, w_1) \leftarrow co(1, i, w_1)$ and $co(j, w_2) \leftarrow co(2, j, w_2)$.

The order $<$ on literals is defined as follows $co(0, w_1) < co(1, w_1) < co(0, w_2) < co(1, w_2) < co(2, w_2) < R$, with R the extended literals $\mathcal{L}^*_{P \cup WC}$ without the *co*/2 atoms. Intuitively, the w_1 constraints are more important than the w_2 constraints, and hence appear below them, and, among each w_i, one rather has a low count than a high count, since this implies less violated constraints. One can check that the preferred answer set of the literal agent $\mathcal{A}$ $=<$ for $P \cup WC$ is $M_3' = M_3 \cup \{c_1^2, co(1, 0, w_1), co(1, 1, w_2), co(2, 1, w_2), co(0, w_1), co(1, w_2)\}$.

Formally, we have the following result, where the weak constraints in a W_j are assumed to be numbered and explicitly tagged with a superscript identifying W_j, i.e. $W_j = \{\leftarrow \beta_1^j, \ldots, \leftarrow \beta_n^j\}$.

Theorem 9. *Let $\langle P, W = \{W_1, \ldots, W_n\}\rangle$ be a weak logic program. M is a preferred answer set of $\langle P, W\rangle$ iff, for all $1 \leq j \leq n$,*

$$\begin{aligned} M' = M & \cup \{c_i^j \mid M \models \beta_i^j\} \cup \{co(1, \alpha, w_j) \mid c_1^j \notin M' \Rightarrow \alpha = 0, c_1^j \in M' \Rightarrow \alpha = 1\} \\ & \cup \{co(k+1, \alpha, w_j) \mid co(k, l, w_j) \in M', 0 \leq l \leq k < |W_j| \wedge \\ & \qquad [c_{k+1}^j \notin M' \Rightarrow \alpha = l, c_{k+1}^j \in M' \Rightarrow \alpha = l+1]\} \\ & \cup \{co(m, w_j) \mid co(|W_j|, m, w_j) \in M'\} \end{aligned}$$

is a preferred answer set of the literal agent $\mathcal{A}$ for $P \cup WC$ where WC, for all $1 \leq j \leq n$, consists of the rules $c_i^j \leftarrow \beta_i^j$, $co(1, 0, w_j) \leftarrow not\ c_1^j$, $co(1, 1, w_j) \leftarrow c_1^j$ and

$$\begin{aligned} co(k+1, l, w_j) &\leftarrow co(k, l, w_j), not\ c_{k+1}^j \text{ with } 0 \leq l \leq k < |W_j| \\ co(k+1, l+1, w_j) &\leftarrow co(k, l, w_j), c_{k+1}^j \end{aligned}$$

together with the rules $co(m, w_j) \leftarrow co(|W_j|, m, w_j)$*,* $0 \leq m \leq |W_j|$*, and the order* $<_{\mathcal{A}}$ *defined as* $co(0, w_1) <_{\mathcal{A}} co(1, w_1) <_{\mathcal{A}} \ldots <_{\mathcal{A}} co(|W_1|, w_1) <_{\mathcal{A}} \ldots <_{\mathcal{A}} co(0, w_n) <_{\mathcal{A}} co(1, w_n) <_{\mathcal{A}} \ldots <_{\mathcal{A}} co(|W_n|, w_n)$*.*

A more general approach, in the spirit of rule and literal agents, is to allow agents to prefer the satisfaction of certain weak constraints over the satisfaction of other ones. A weak logic program then becomes a pair $\langle P, W\rangle$, where P is a program and W is a set of constraints. A *weak agent* for $\langle P, W\rangle$ corresponds to a well-founded strict partial order on W, which induces an order $\sqsubseteq$ among interpretations M and N of P such that, $M \sqsubseteq N$ iff $\forall w_2 \in W_N \backslash W_M \cdot \exists w_1 \in W_M \backslash W_N \cdot w_1 < w_2$, where W_X are the weak constraints in W that are satisfied by X, mirroring Definition 1 for rule agents (note that the latter are different from weak agents since RAs require the satisfaction of all constraints in all extended answer sets).

The extended answer sets of P are, by definition, the extended answer sets of a weak agent $\mathcal{A}$ for $\langle P, W\rangle$, and preferred answer sets are defined as the minimal extended answers sets w.r.t. $\sqsubset$. Note that a preferred answer set M of a weak agent $\mathcal{A}$ for $\langle P, W\rangle$ has a minimal set of violated constraints, i.e. there is no extended answer set N of $\mathcal{A}$ such that $W \backslash W_N \subset W \backslash W_M$.

For a program P, define the extended program $E(P)$ as P with the rules $a \leftarrow \beta$ replaced by $a \leftarrow \beta, not\ \neg a$. From Theorem 4 in [22] we have that the extended answer sets of P are exactly the answer sets of $E(P)$. We can then rewrite a weak agent as a rule agent by introducing for each weak constraint $w :\ \leftarrow \beta$ rules $w \leftarrow \beta$ and $\neg w \leftarrow \beta$ such that w is in an answer set if the constraint is violated.

Theorem 10. *Let* $\mathcal{A}^w$ *be a weak agent for a WLP* $\langle P, W\rangle$*.* M *is a preferred answer set of* $\mathcal{A}^w$ *for* $\langle P, W\rangle$ *iff* $M' = M \cup \{w \mid w \in W, M \not\models w\}$ *is a preferred answer set of the RA* $\mathcal{A}^r$ *for* $E(P) \cup WC$ *with* $WC = \{w \leftarrow \beta; \neg w \leftarrow \beta;\ \leftarrow \beta, not\ w \mid w :\ \leftarrow \beta \in W\}$ *and* $\neg w_1 \leftarrow \beta_1 <_{\mathcal{A}^r} \neg w_2 \leftarrow \beta_2$ *iff* $w_1 <_{\mathcal{A}^w} w_2$*.*

Moreover, weak agents are as expressive as rule agents.

Theorem 11. *Let* P *be a program and* $\mathcal{A}^r$ *a RA for* P*.* M *is a preferred answer set of* $\mathcal{A}^r$ *for* P *iff* M *is a preferred answer set of the weak agent* $\mathcal{A}^w$ *for the WLP* $\langle P, W\rangle$*, with* $W = \{\ \leftarrow \beta, not\ \alpha \mid \alpha \leftarrow \beta \in P\}$ *and* $\leftarrow \beta_1, not\ \alpha_1 <_{\mathcal{A}^w}\ \leftarrow \beta_2, not\ \alpha_2$ *iff* $\alpha_1 \leftarrow \beta_1 <_{\mathcal{A}^r} \alpha_2 \leftarrow \beta_2$*.*

Weak agents, placed in a hierarchy, then allow for an intuitive decision making process based on satisfaction and violation of weak constraints. The complexity of weak agent hierarchies can easily be deduced from the reductions from and to rule agent hierarchies, with Theorem 10 and 11 and their extensions for hierarchies.

Theorem 12. *The problem of deciding, given a weak agent hierarchy* $\langle\langle P, W\rangle, T\rangle$ *and a literal* l*, whether there is a preferred answer set containing* l *is* $\Sigma_{d(T)+2}$*-complete. On the other hand, deciding whether every preferred answer set contains* l *is* $\Pi_{d(T)+2}$*-complete.*

5 Conclusions and Directions for Further Research

In this paper, we introduced a system suitable to model hierarchical decision making. We equip agents with a preference relation on the available knowledge and allow them to cooperate with each other in a hierarchical fashion. Preferred solutions of these systems naturally correspond to preferred decisions regarding the problem.

Initially, we defined two types of preference agents: rule agents express a preference over rules, while literal agents use a preference over extended literals they rather prefer upon others in a solution. We showed that mixed AHs, containing both types of agents, can be reduced to hierarchies consisting only of rule or literal agents. It turns out that these AHs cover the polynomial hierarchy.

Finally, we showed that layered weak constraints can be easily simulated by a single agent. Furthermore, we generalized the concept of layered weak constraints to weak agent hierarchies, which are equivalent to rule agent hierarchies.

Future work comprises a dedicated implementation of the approach, using existing answer set solvers. E.g., we could generate an extended answer set which is then improved recursively by a set of augmented programs, corresponding to the agents in the hierarchy, generating strictly better solutions. A fixpoint of this procedure then corresponds to a preferred answer set of the system.

References

1. José Júlio Alferes and Luís Moniz Pereira. Updates plus preferences. In Manual Ojeda-Aciego, Inma P. de Guzmán, Gerhard Brewka, and Luíz Moniz Pereira, editors, *European Workshop, JELIA 2000*, volume 1919 of *Lecture Notes in Artificial Intelligence*, pages 345–360, Malaga, Spain, September–October 2000. Springer Verlag.
2. Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge Press, 2003.
3. Chitta Baral and Michael Gelfond. Reasoning agents in dynamic domains. In *Logic-based artificial intelligence*, pages 257–279. Kluwer Academic Publishers, 2000.
4. G. Brewka. Logic programming with ordered disjunction. In *Proceedings of the 18th National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence*, pages 100–105, Edmonton, Canada, July 2002. AAAI Press.
5. Gerhard Brewka and Thomas Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356, April 1999.
6. Gerhard Brewka, Ilkka Niemela, and Tommi Syrjanen. Implementing ordered disjunction using answer set solvers for normal programs. In Flesca et al. [14], pages 444–455.
7. Francesco Buccafurri, Wolfgang Faber, and Nicola Leone. Disjunctive logic programs with inheritance. In Danny De Schreye, editor, *Logic Programming: The 1999 International Conference*, pages 79–93, Las Cruces, New Mexico, December 1999. MIT Press.
8. Francesco Buccafurri and Georg Gottlob. Multiagent compromises, joint fixpoints, and stable models. In Antonis C. Kakas and Fariba Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, volume 2407 of *Lecture Notes in Computer Science*, pages 561–585. Springer, 2002.
9. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Strong and weak constraints in disjunctive datalog. In *Proceedings of the 4th International Conference on Logic Programming (LPNMR '97)*, pages 2–17, 1997.
10. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Disjunctive ordered logic: Semantics and expressiveness. In Anthony G. Cohn, Lenhard K. Schubert, and Stuart C. Shapiro, editors, *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning*, pages 418–431, Trento, June 1998. Morgan Kaufmann.

11. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Enhancing disjunctive datalog by constraints. *Knowledge and Data Engineering*, 12(5):845–860, 2000.
12. Wolfgang Faber. Disjunctive datalog with strong and weak constraints: Representational and computational issues. Master's thesis, Institut for Informationssysteme, Technische Universität Wien, 1998.
13. Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. Representing school timetabling in a disjunctive logic programming language. In *Proceedings of the 13th Workshop on Logic Programming (WLP '98)*, 1998.
14. Sergio Flesca, Sergio Greco, Nicola Leone, and Giovambattista Ianni, editors. *European Conference on Logics in Artificial Intelligence (JELIA '02)*, volume 2424 of *Lecture Notes in Artificial Intelligence*, Cosenza, Italy, September 2002. Springer Verlag.
15. D. Gabbay, E. Laenens, and D. Vermeir. Credulous vs. Sceptical Semantics for Ordered Logic Programs. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 208–217, Cambridge, Mass, 1991. Morgan Kaufmann.
16. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, pages 1070–1080, Seattle, Washington, August 1988. The MIT Press.
17. Robert A. Kowalski and Fariba Sadri. Logic programs with exceptions. In David H. D. Warren and Peter Szeredi, editors, *Proceedings of the 7th International Conference on Logic Programming*, pages 598–613, Jerusalem, 1990. The MIT Press.
18. Els Laenens and Dirk Vermeir. A logical basis for object oriented programming. In Jan van Eijck, editor, *European Workshop, JELIA 90*, volume 478 of *Lecture Notes in Artificial Intelligence*, pages 317–332, Amsterdam, The Netherlands, September 1990. Springer Verlag.
19. Davy Van Nieuwenborgh, Stijn Heymans, and Dirk Vermeir. On programs with linearly ordered multiple preferences, 2004. Accepted at ICLP '04.
20. Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.
21. Chiaki Sakama and Katsumi Inoue. Representing priorities in logic programs. In Michael J. Maher, editor, *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming*, pages 82–96, Bonn, September 1996. MIT Press.
22. Davy Van Nieuwenborgh and Dirk Vermeir. Preferred answer sets for ordered logic programs. In Flesca et al. [14], pages 432–443.
23. Davy Van Nieuwenborgh and Dirk Vermeir. Order and negation as failure. In Catuscia Palamidessi, editor, *ICLP*, volume 2916 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2003.
24. Davy Van Nieuwenborgh and Dirk Vermeir. Ordered diagnosis. In *Proceedings of the 10th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR2003)*, volume 2850 of *Lecture Notes in Artificial Intelligence*, pages 244–258, Almaty, Kazachstan, 2003. Springer Verlag.
25. Davy Van Nieuwenborgh and Dirk Vermeir. Ordered programs as abductive systems. In *Proceedings of the APPIA-GULP-PRODE Conference on Declarative Programming (AGP2003)*, pages 374–385, Regio di Calabria, Italy, 2003.
26. Marina De Vos and Dirk Vermeir. Logic programming agents playing games. In *Research and Development in Intelligent Systems XIX (ES2002)*, BCS Conference Series, pages 323–336. Springer-Verlag, 2002.
27. Kewen Wang, Lizhu Zhou, and Fangzhen Lin. Alternating fixpoint theory for logic programs with priority. In *Proceedings of the First International Conference on Computational Logic (CL2000)*, volume 1861 of *Lecture Notes in Computer Science*, pages 164–178, London, UK, July 2000. Springer.