An Ordered Choice Logic Programming Front-End for Answer Set Solvers

Marina De Vos*

Department of Computer Science
University of Bath
Bath, United Kingdom
mdv@bath.ac.uk

Abstract. Ordered Choice Logic Programming (OCLP) allows for preference-based decision-making with multiple alternatives without the burden of any form of negation. This complete absence of negation does not weaken the language as both forms (classical and as-failure) can be intuitively simulated in the language. The semantics of the language is based on the preference between alternatives, yielding both a skeptical and a credulous approach. In this paper we demonstrate how OCLPs can be translated to semi-negative logic programs such that, depending on the transformation, the answer sets of the latter correspond with the skeptical or credulous answer sets of the former. By providing such a mapping, we have a mechanism for implementing OCLP using an answer set solver like smodels or dlv.

1 Introduction

Examining human reasoning, we find that people often use preference, order or defaults for making decisions: “I prefer this dish”, “This color goes better with the interior”, “This item costs more”, “In general, the human heart is positioned at the left”. When faced with conflicting information, one tends to make decisions that prefer an alternative corresponding to more reliable, more complete, more preferred or more specific information. When modeling knowledge or non-monotonic reasoning via computer programs, it is only natural to incorporate such mechanisms.

In recent years several proposals for the explicit representation of preference in logic programming formalisms have been put forward. [11, 10] are just two examples.

Systems that support preferences find applications in various domains such as law, object orientation, scheduling, model based diagnosis and configuration tasks. However, most approaches use preferences only when the models have already been computed, i.e. decisions have already been made, or only support preferences between rules with opposite (contradictory) consequences, thus statically limiting the number of alternatives of a decision.

* 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.

In [8], we proposed a formalism, called ordered choice logic programming, that enables one to dynamically reason about situation-dependent decisions involving multiple alternatives. The dynamics of this system is demonstrated by the following example.

Example 1. Buying a laptop computer involves a compromise between what is desirable and what is affordable. Take, for example, the choice between a CD, CDRW or DVD drive. The CD is the cheaper option. On the other hand, for a laptop, a DVD drive may be more useful than a CD writer. If the budget is large enough, one could even buy two of the devices. The above information leads one to consider two possible situations.

- With a smaller budget, a DVD-player is indicated, while
- with a larger budget, one can order both a DVD-player and a CD-writer.

To allow this kind of reasoning, a program consists of a (strict) partially ordered set of components containing choice rules (rules with exclusive disjunction in the head). Information flows from less specific components to the more preferred ones until a conflict among alternatives arises, in which case the most specific one will be favored. The situation becomes less clear when two alternatives are equally valued or are unrelated. The decision in this case is very situation dependent: a doctor having a choice between two equally effective cures has to make a decision, while you better remain indecisive when two of your friends have an argument! To allow both types of intuitive reasoning, [8] introduces both a credulous and skeptical semantics.

OCLP provides an elegant and intuitive way of representing and dealing with decisions. People with little or no experience with non-monotonic reasoning can easily relate to it, due to the absence of negation. This absence of negation does not restrict the language in any way, as both types of negation (classic and as-failure) can easily be simulated ([8]).

In computer science, having a nice theory alone is not enough; one also needs to be able to apply it. The aim of this paper is to provide the theoretical foundations for it.

In this paper, we investigate the possibility for building an OCLP front-end for answer set solvers. Smodels ([12]), developed at Helsinki University of Technology, and dlv ([17]), created at the Technical University of Vienna and the University of Calabria are currently the most popular ones.

The remainder of this paper is organized as follows: we continue in Section2 with short overview of the basic information concerning choice logic programming, the language behind OCLP. Section 3 focuses on the introduction of OCLP with its skeptical and credulous answer set semantics. Section 4 deals with a mapping of OCLP to semi-negative logic programs allowing answer set solvers to work with OCLP. These mappings, one for each semantics, can then serve as the foundations on which we build the OCLP front-end. We end this paper with a discussion on the relations to other approaches (Section 5) and directions for future research (Section 6).

2 Choice Logic Programming

Choice logic programs [7] represent decisions by interpreting the head of a rule as an exclusive choice between alternatives.

Formally, a *Choice Logic Program* [7], CLP for short, is a countable set of rules of the form $A \leftarrow B$ where A and B are finite sets of ground atoms. Intuitively, atoms in A are assumed to be xor'ed together while B is read as a conjunction (note that A may be empty, i.e. constraints are allowed). The set A is called the head of the rule r, denoted H_r, while B is its body, denoted B_r. In examples, we use "$\oplus$" to denote exclusive disjunction, while "," is used to denote conjunction.

The *Herbrand base* of a CLP P, denoted $\mathcal{B}_P$, is the set of all atoms that appear in P. An *interpretation*[1] is a subset of $\mathcal{B}_P$.

A rule r in a CLP is said to be *applicable* w.r.t. an interpretation I if $B_r \subseteq I$. Since we are modeling choice, we have that r is *applied* when r is applicable and $|H_r \cap I| = 1$[2]. A rule is *satisfied* if it is applied or not applicable. A *model* is defined in the usual way as a total interpretation that satisfies every rule. A model M is said to be *minimal* if there does not exist a model N such that $N^+ \subset M^+$.

3 Ordered Choice Logic Programming

An ordered choice logic program (OCLP) is a collection of choice logic programs, called components, which are organized in a strict partial order[3] that represents some preference criterion (e.g. specificity, reliability, ...).

Definition 1. *An **Ordered Choice Logic Program**, or OCLP, is a pair $\langle \mathcal{C}, \prec \rangle$ where $\mathcal{C}$ is a finite set of choice logic programs, called **components**, and "$\prec$" is a strict pointed partial order on $\mathcal{C}$.*

For two components $C_1, C_2 \in \mathcal{C}$, $C_1 \prec C_2$ implies that C_1 is preferred over C_2. Throughout the examples, we will often represent an OCLP P by means of a directed acyclic graph (dag) in which the nodes represent the components and the arcs the $\prec$-relation, where arcs point from smaller (more preferred) to larger (less preferred) components.

Example 2. The decision problem from the introduction (Example 1) can easily be written as an OCLP, as shown in Figure 1. The rules in components P_1, P_2 and P_3 express the preferences in case of a small budget. The rules in P_4 express the intention to buy/configure a laptop and, because of this, a decision about its various devices should be made. In component P_5, the first rule states the possibility of a larger budget. If so, the two remaining rules allow the purchase of both a DVD-player and a CD-writer.

Definition 2. *Let P be an OCLP. We use $P^\star$ to denote the CLP that contains all the rules appearing in (a component of) P. We assume that rules in $P^\star$ are labeled by the component from which they originate and we use $c(r)$ to denote the component of r*[4].

[1] In this paper we only work with total interpretations: each atom from the Herbrand base is either true or false. Bearing this in mind, it suffices to mention only those atoms which can be considered true.

[2] For a a set X, we use $|X|$ do denote its cardinality.

[3] A relation R on a set A is a strict partial order iff R is anti-reflexive, anti-symmetric and transitive. R is pointed if an element $a \in A$ exists such that aRb for all $b \in A$.

[4] Without losing generality, we can assume that a rule appears in only one component.

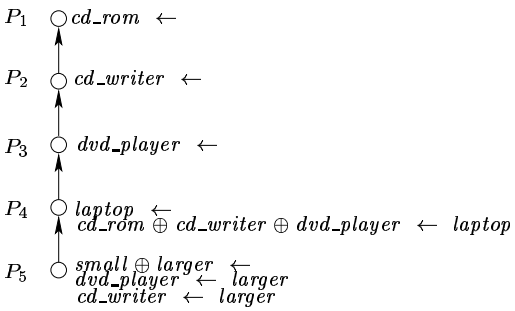

Fig. 1. The Configuration OCLP of Example 2

The Herbrand base $\mathcal{B}_P$ *of* P *is defined by* $\mathcal{B}_P = \mathcal{B}_{P^\star}$.
An ***interpretation*** *for* P *is any interpretation of* $P^\star$. *We say that a rule* r *in* P *is* ***applicable*** *w.r.t. an interpretation* I *iff* $B_r \subseteq I$; r *is* ***applied*** *w.r.t.* I *iff* r *is applicable and* $|H_r \cap I| = 1$.

Example 3. For the OCLP in Example 2, the sets $I = \{dvd\_player, small\}$, $J = \{laptop, cd\_writer, small\}$, $K = \{laptop, dvd\_player, small\}$ and $L = \{dvd\_player, larger, cd\_writer, cd\_player, laptop\}$ are all interpretations. The interpretation I makes the rule $small \oplus larger \leftarrow$ applied while the applicable rule $cd\_writer \leftarrow$ is not applied.

Facing a decision means making an exclusive choice between the various alternatives which are available. If we want OCLP to model/solve decision problems we need a mechanism for representing them. In a CLP, decisions are generated by so-called *choice rules* i.e. rules with multiple head atoms. For OCLP, we can do a something similar as long as we also take the preference order into account. We want to make sure that we leave the option open to overrule the exclusiveness of a choice when in more preferred components multiple alternatives are suggested (e.g. Example 1). Hence we say that an atom a is an *alternative* for an atom b in a component C if an applicable rule exists in a component at least as preferred as C containing both a and b in its head.

Definition 3. *Let* I *be an interpretation of an OCLP* $P = \langle \mathcal{C}, \prec \rangle$ *with* $C \in \mathcal{C}$. *The set of* ***alternatives*** *in* C *for an atom* $a \in \mathcal{B}_P$ *w.r.t.* I, *denoted* $\Omega^I_C(a)$, *is defined as*[5]*:* $\Omega^I_C(a) = \{b \mid \exists r \in P^\star \cdot c(r) \preccurlyeq C \ \wedge \ B_r \subseteq I \ \wedge \ a, b \in H_r \text{ with } a \neq b\}$.

Example 4. Reconsider Example 3. The alternatives for cd_rom in P_2 w.r.t. J are $\Omega^J_{P_2}(cd\_rom) = \{dvd\_player, cd\_writer\}$. W.r.t. I, we obtain $\Omega^I_{P_2}(cd\_rom) = \emptyset$, since the choice rule in P_4 is not applicable. When we take P_5 instead of P_2, we obtain w.r.t. J: $\Omega^J_{P_5}(cd\_rom) = \emptyset$.

Given the alternatives in a certain context (a component and an interpretation), one naturally selects that alternative that is motivated by a more preferred rule, thus *defeating* the rule(s) suggesting less preferred alternatives. However, if alternatives appear

[5] $\preccurlyeq$ is the reflexive closure of $\prec$.

in the same or unrelated components, two approaches are possible: using a skeptical strategy, one would refrain from making a decision, i.e. not selecting any of the various alternatives, while a credulous setting suggests an arbitrary choice of one of the alternatives. For both types of reasoning one can think of situations where one approach works while the other gives an incorrect, unintuitive outcome. Skeptical reasoning is practiced in American law when a jury cannot come to a unanimous decision. An example of credulous reasoning is the decision a goal-keeper faces in football when trying to stop a penalty. To accommodate this problem, we introduce a semantics for both types of reasoning. From a skeptical viewpoint, we say that rule is defeated if one can find a better, more preferred alternative for each of its head atoms.

Definition 4. *Let I be an interpretation for an OCLP P. A rule $r \in P^\star$ is **defeated** w.r.t. I iff* $\forall a \in H_r \cdot \exists r' \in P^\star \cdot c(r') \prec c(r) \ \wedge \ B_{r'} \subseteq I \ \wedge \ H_{r'} \subseteq \Omega^I_{c(r)}(a)$.

Example 5. Reconsider Example 3. The rule $cd\_rom \leftarrow$ is defeated w.r.t. J by the rule $cd\_writer \leftarrow$. The rule $cd\_rom \oplus cd\_writer \oplus dvd\_player \leftarrow$ is defeated w.r.t. L by the combination of the rules $dvd\_player \leftarrow larger$ and $cd\_writer \leftarrow larger$.

Example 6. Consider the OCLP $\langle \{P_1 = \{a \leftarrow ; b \leftarrow \}, P_2 = \{a \oplus b \leftarrow \}\}, P_2 \prec P_1 \rangle$. Given the interpretation $\{b\}$, the rule $a \leftarrow$ is not defeated as the only alternative of a, i.e. b, is not brought forward in a more preferred component.

Just as for the skeptical semantics we need to define an appropriate defeating strategy. An obvious way of doing so consists of simply dropping the condition that an alternative should be found in a more preferred component. Unfortunately, this leads to unintuitive results. To avoid this, we need to make sure that credulous defeaters are not only applicable, but also applied.

Definition 5. *Let I be an interpretation for an OCLP P. A rule $r \in P^\star$ is **c-defeated** w.r.t. I iff* $\forall a \in H_r \cdot \exists r' \in P^\star \cdot c(r) \not\prec c(r') \wedge r'$ *is applied w.r.t.* $I \wedge H_{r'} \subseteq \Omega^I_{c(r)}(a)$.

Example 7. While the skeptical approach makes it impossible to have the rule $a \leftarrow$ in Example 6 defeated w.r.t. $\{b\}$, the credulous semantics can.

For our model semantics, both skeptical as credulous, rules that are not satisfied (as for choice logic programs) must be (c-)defeated.

Definition 6. *Let P be an OCLP. A total interpretation I is a **skeptical/credulous model** iff every rule in $P^\star$ is either not applicable, applied or (c-)defeated w.r.t. I. A skeptical/credulous model M is **minimal** iff M is minimal according to set inclusion, i.e. no skeptical/credulous model N of P exists such that $N^+ \subset M^+$.*

Example 8. Reconsider the interpretations I, J, K and L from Example 3. Only K and L are skeptical/credulous models. Model L is not minimal due to the skeptical/credulous model $Z = \{dvd\_player, cd\_writer, laptop, larger\}$. The minimal skeptical/credulous models K and Z correspond to the intuitive outcomes of the problem.

Example 9. The program of Example 6 has no skeptical models but two credulous ones: $\{a\}$, $\{b\}$.

The next example illustrates that the skeptical/credulous model semantics does not always provide the appropriate solutions to the decision problem at hand.

Example 10. Consider the ordered choice logic program $P = \langle\{P_1 = \{a \leftarrow\}, P_2 = \{b \leftarrow\}, P_3 = \{a \oplus b \leftarrow c\}, P_3 \prec P_2 \prec P_1\rangle$, where P has two minimal skeptical/credulous models: $M = \{b, c\}$, and $N = \{a, b\}$. Clearly, c is an unsupported assumption in M, causing P_3 to trigger an unwarranted choice between a and b.

We introduce an adaptation of the Gelfond-Lifschitz [14] and reduct ([16]) transformations to filter unintended (minimal) models containing unsupported atoms. This results in the skeptical/credulous answer set semantics.

Definition 7. *Let M be a total interpretation for an OCLP P. The **Gelfond-Lifschitz transformation** (resp. **reduct**) for P w.r.t. M, denoted P^M (resp. P_c^M), is the CLP obtained from $P^\star$ by removing all (c-)defeated rules. M is called a **skeptical (resp. credulous) answer set** for P iff M is a minimal model*[6] *for P^M (resp. P_c^M).*

Although both answer set semantics produce models (skeptical or credulous ones) for the program, they differ in whether they produce minimal ones or not. Just as for answer sets of semi-negative logic programs, we find that skeptical answer sets are minimal skeptical models. For extended disjunctive logic programs, the answer set semantics is not minimal[16]. The same applies for credulous answer sets of ordered choice logic programs, as demonstrated by the following example.

Example 11. Consider the program $P = \langle\{P_1 = \{r_1 : g \leftarrow\}, P_2 = \{r_2 : p \oplus d \leftarrow; r_3 : g \oplus p \leftarrow; r_4 : g \oplus d \leftarrow\}\}, P_2 \prec P_1\rangle$. Consider $M_1 = \{g\}$ and $M_2 = \{g, d\}$. Clearly, $M_1^+ \subset M_2^+$, while both interpretations are credulous answer sets for P. For M_1, we have that $P_c^{M_1} = \{g \leftarrow; g \oplus d \leftarrow; g \oplus p \leftarrow\}$ for which it can easily be verified that M_1 is a minimal model. The program $P_c^{M_2} = \{p \oplus d \leftarrow ; g \oplus p \leftarrow\}$ has two minimal models: $\{p\}$ and $\{g, d\}$. Note that M_2 is a credulous model because the c-defeater has become c-defeated, i.e. the justification in M_1 for c-defeating $p \oplus d \leftarrow$ has disappeared in M_2.

Non-minimal credulous answer sets appear when the program contains inconsistencies on a decision level: in the above example the following choices have to be made: $\{p, d\}$, $\{g, p\}$ and $\{g, d\}$. Because of the program's construction, one can choose either one or two alternatives and c-defeating will make the choice justifiable.

4 Implementation

For the last five years, answer set programming has gained popularity. One of the main forces behind this is the growing efficiency of answer solvers like smodels ([12]) and dlv ([17]).

In this section, we propose a mapping, for both semantics, to semi-negative logic programs. Since both answer set solvers support this type of programs, this would allow

[6] The definition in [8] states a stable model, but since both are identical for CLP, we have opted in this paper to use minimal model instead.

us to implement an OCLP front-end for them. Due to page restrictions, we are unable to provide proofs for the presented theorems. However, full details can be found in the technical appendix to this paper which is accessible from:
http://www.cs.bath.ac.uk/∼mdv/publications.html.

4.1 Skeptical Mapping

The skeptical answer set semantics is based on the notion of defeat. If we want to map our formalism to a language which does not support this, we need a way to encode it. This implies anticipating which combinations of rules could be capable of defeating a rule and which ones are not.

The definition of defeating relies strongly on the notion of alternatives: rules can only be defeated by rules containing alternatives of the head atoms. Therefore, anticipating defeaters also implies predicting alternatives. According to Definition 3, b is an alternative of a in a component C if one can find an applicable choice rule as preferred as C containing both a and b in the head. This implies that even without an interpretation we can find out which atoms might be alternatives; it only remains to be checked if the rule is applicable or not. These condition-based alternatives are called a possible future alternatives and are defined more formally below.

Definition 8. *Let P be an OCLP, $C \in \mathcal{C}$ be component of P and $a \in \mathcal{B}_P$. The set of **possible future alternatives** of a in C, denoted as $\mathcal{A}^P_C(a)$, is defined as $\mathcal{A}^P_C(a) = \{(b, B_r) \mid \exists r \in P \cdot c(r) \preccurlyeq C, a, b \in H_r, a \neq b\}$.*

Example 12. Consider the OCLP $P = \langle\{P_1 = \{r_1 : a \leftarrow; r_2 : f \leftarrow\}, P_2 = \{r_3 : a \oplus b \oplus c \leftarrow d; r_4 : a \oplus d \leftarrow f; r_5 : d \oplus c \leftarrow\}, P_2 \prec P_1\}\rangle$. The possible future alternatives of a in P_1 equal $\mathcal{A}^P_{P_1}(a) = \{(b, \{d\}), (c, \{d\}), (d, \{f\})\}$.

The next theorem demonstrates that alternatives can be expressed in terms of possible future alternatives.

Theorem 1. *Let P be an OCLP, $C \in \mathcal{C}$ be component of P, $a \in \mathcal{B}_P$ and I an interpretation for P. Then, $\Omega^I_C(a) = \{b \mid (b, S) \in \mathcal{A}^P_C(a) \wedge S \subseteq I\}$.*

Having these possible future alternatives allows us to detect possible future defeaters in much the same way as we detect standard defeaters (Definition 4). The only extra bit we need is to collect all the conditions on the alternatives. This collection then acts as the condition for the defeating rule.

Definition 9. *Let P be an OCLP, $C \in \mathcal{C}$ be component of P and $a \in \mathcal{B}_P$. The set of **possible future defeaters** of a in C, denoted as $\mathcal{D}^P_C(a)$, is defined as $\mathcal{D}^P_C(a) = \{(r, S) \mid \exists r \in P \cdot c(r) \prec C, \forall b \in H_r \cdot \exists (b, B_b) \in \mathcal{A}^P_C(a) \cdot S = \bigcup B_b \cup B_r\}$.*

Note that possible future defeaters are defined for atoms and not for rules. This makes the transformation easier to read and unclutters the definitions.

Example 13. When we look back to the program P of Example 12, we have that a has a one possible future defeater in P_1 as: $\mathcal{D}^P_{P_1}(a) = \{(r_5, \{d, f\})\}$. All the other atoms in the program do not have any possible future defeaters in any of the components.

Clearly, possible future defeaters can be used for expressing interpretation-dependent defeaters.

Theorem 2. *Let P be an OCLP and let I be an interpretation for it. A rule $r \in P^\star$ is defeated w.r.t. I iff $\forall a \in H_r \cdot \exists (r', S) \in \mathcal{D}^P_{c(r)}(a) \cdot B_{r'} \subseteq I, S \subseteq I$.*

These possible future defeaters are the key to mapping OCLPs to semi-negative logic programs. We are only required to turn the information which makes possible future defeaters into defeaters, i.e. they have to be applicable, into a condition. To make this possible, we introduce for each non-constraint rule r in the program two new atoms: d_r and a_r. The former indicates that the rule r is defeated or not, while the truth value of the latter is an indicator of the applicability of the rule.

Definition 10. *Let P be an OCLP. Then, the logic program $P_\neg$ is defined as follows:*

1. $|H_r| = 0$*:* $r \in P_\neg$
2. $|H_r| \geq 1$*:*
   - *(a)* $a \leftarrow B_r, \neg d_r \neg (H_r \setminus \{a\}) \ \in P_\neg$*:* $\forall a \in H_r$
   - *(b)* $a_r \leftarrow B_r \ \in P_\neg$
   - *(c)* $d_r \leftarrow C \ \in P_\neg$*:* $\forall a \in H_r \cdot \exists (r_a, S_a) \in \mathcal{D}^P_{c(r)}(a) \cdot C = \bigcup (a_{r_a} \cup S_a)$
   - *(d)* $\leftarrow a, b, B_r, \neg d_r \ \in P_\neg$*:* $\forall a, b \in H_r \cdot a \neq b$

Since constraints are not involved in the defeating process, we can simply copy them to the corresponding logic program. For the answer set semantics of ordered choice logic program, we need, among other things, that each applicable, undefeated rule admits exactly one head atom. Rules of type a) and d) make sure that the corresponding rules in the logic program do not viololate this property. The rules of type b) indicate which orginal rules are applicable. The c)-rules are probably the most difficult ones. They express when a rule should or could be considered defeated. If we look at Theorem 2, we have a mechanism for relating possible future defeaters to actual defeaters. For defeaters we need an interpretation, but if we put the set of atoms that need to be true in the body of a rule simulating defeat, we can be sure that models making these rules applicable, can be used to defeat the original rule. Thanks to rules of type b), we can replace those elements with a_r

Example 14. The corresponding logic program $P_\neg$ of the OCLP of Example 12 looks like:

$a \leftarrow \neg d_{r_1}$	$a \leftarrow f, \neg d, \neg d_{r_4}$	$a_{r_2} \leftarrow$	$\leftarrow d, \neg d_{r_3}, a, b$
$f \leftarrow \neg d_{r_2}$	$d \leftarrow f, \neg a, \neg d_{r_4}$	$a_{r_3} \leftarrow d$	$\leftarrow d, \neg d_{r_3}, a, c$
$a \leftarrow d, \neg b, \neg c, \neg d_{r_3}$	$d \leftarrow \neg c, \neg d_{r_5}$	$a_{r_4} \leftarrow f$	$\leftarrow d, \neg d_{r_3}, b, c$
$b \leftarrow d, \neg a, \neg c, \neg d_{r_3}$	$c \leftarrow \neg d, \neg d_{r_5}$	$a_{r_5} \leftarrow$	$\leftarrow f, \neg d_{r_4}, a, c$
$c \leftarrow d, \neg a, \neg b, \neg d_{r_3}$	$a_{r_1} \leftarrow$	$d_{r_1} \leftarrow a_{r_5}, d, f$	$\leftarrow \neg d_{r_5}, d, c$

The original OCLP of Example 12 has two skeptical answer sets, $\{f, d\}$ and $\{f, c, a\}$, which correspond exactly with the two answer sets, $\{a_{r_1}, a_{r_2}, a_{r_3}, a_{r_4}, a_{r_5}, d_{r_1}, f, d\}$ and $\{a_{r_1}, a_{r_2}, a_{r_4}, a_{r_5}, f, c, a\}$, of $P_\neg$.

Theorem 3. *Let P be an OCLP and $P_\neg$ be its corresponding logic program. Then, a one-to-one mapping exists between the skeptical answer sets M of P and the answer sets N of $P_\neg$ in such a way that $N = M \cup \{a_r \mid \exists r \in P \cdot |H_r| \geq 1, B_r \subseteq M\} \cup \{d_r \mid \exists r \in P \cdot r$ is defeated w.r.t. $M\}$.*

4.2 Credulous Mapping

To obtain the credulous answer set semantics for OCLPs, we propose a similar mapping to semi-negative logic programs. The only difference between the skeptical and the credulous semantics is the way they both handle defeat. For the credulous version, we need to make sure that we look for c-defeaters in all components which are not less preferred as the rule we wish to defeat. Furthermore, we have to make sure that c-defeaters are applied and not just applicable as is the case for defeaters. The former will be encoded by means of possible future c-defeaters while the latter will be translated in a different style of a_r rules in the mapping.

The definition of possible future c-defeater is identical to the one of its skeptical counter-part except that it looks for rules in all components which are not less preferred.

Definition 11. *Let P be an OCLP, $C \in \mathcal{C}$ be component of P and $a \in \mathcal{B}_P$. The set of **possible future c-defeaters** of a in C, denoted as $\mathcal{F}_C^P(a)$, is defined as $\mathcal{F}_C^P(a) = \{(r, S) \mid \exists r \in P \cdot C \not\prec c(r), \forall b \in H_r \cdot \exists (b, B_b) \in \mathcal{A}_C^P(a) \cdot S = \bigcup B_b\}$.*

Just as before, c-defeaters can be expressed in terms of possible future c-defeaters.

Theorem 4. *Let P be an OCLP and let I be an interpretation for it. A rule $r \in P^\star$ is c-defeated w.r.t. I iff $\forall a \in H_r \cdot \exists (r', S) \in \mathcal{D}_{c(r)}^P(a) \cdot B_{r'} \subseteq I, S \subseteq I$.*

Definition 12. *Let P be an OCLP. Then, the logic program $P_\neg^c$ is defined as follows:*

1. $|H_r| = 0$*:* $r \in P_\neg^c$
2. $|H_r| \geq 1$*:*
   - *(a)* $a \leftarrow B_r, \neg d_r, \neg(H_r \setminus \{a\}) \in P_\neg^c$*:* $\forall a \in H_r$
   - *(b)* $a_r \leftarrow B_r, a, \neg(H_r \setminus \{a\}) \in P_\neg^c$*:* $\forall a \in H_r$
   - *(c)* $d_r \leftarrow C \in P_\neg$*:* $\forall a \in H_r \cdot \exists (r_a, S_a) \in \mathcal{F}_{c(r)}^P(a) \cdot C = \bigcup (a_{r_a} \cup S_a)$

The credulous mapping is very similar to the skeptical one but there are a couple of subtle differences: an obvious difference is the use of possible future c-defeater instead of their skeptical counterparts (c-rules). The second change are the rules implying a_r (b-rules). Previously they were used to indicate applicability, the necessary condition for the defeat. Since c-defeat works with applied defeaters, we need to make sure that a_r is considered only true when r is applied. The less obvious change is the absence of the rules of type d). Since a rule can only be applied when one and only one head atom is considered true and because a_r should only be considered true in this particular case, they no longer necessary.

Example 15. Reconsider the OCLP from Example 11. If we use the mapping from Definition 12, we obtain the following program:

$$
\begin{array}{lll}
g \leftarrow \neg d_1 & a_1 \leftarrow g & d_1 \leftarrow a_2 \\
p \leftarrow \neg d, \neg d_2 & a_2 \leftarrow p, \neg d & d_2 \leftarrow a_3, a_4 \\
d \leftarrow \neg p, \neg d_2 & a_2 \leftarrow d, \neg p & d_3 \leftarrow a_2, a_4 \\
g \leftarrow \neg p, \neg d_3 & a_3 \leftarrow g, \neg p & d_4 \leftarrow a_2, a_3 \\
p \leftarrow \neg g, \neg d_3 & a_3 \leftarrow p, \neg g & \\
g \leftarrow \neg d, \neg d_4 & a_4 \leftarrow g, \neg d & \\
d \leftarrow \neg g, \neg d_4 & a_4 \leftarrow d, \neg g &
\end{array}
$$

The answer sets of this program correspond perfectly to the credulous answer sets of the original program. The newly introduced atoms make sure that the answer set semantics remains minimal while the credulous OCLP version is clearly not.

Theorem 5. *Let P be an OCLP and $P_\neg$ be its corresponding logic program. Then, a one-to-one mapping exists between the credulous answer sets M of P and the answer sets N of $P_\neg$ in such a way that $N = M \cup \{a_r \mid \exists r \in P \cdot |H_r| \geq 1, B_r \subseteq M, |H_r \cap M| = 1\} \cup \{d_r \mid \exists r \in P \cdot r$ is c-defeated w.r.t. $M\}$.*

5 Relationship to Other Approaches

Our formalism shows similarities with ordered logic programming[13, 15, 5], where the latter supports disjunction (in the head), which also provides a skeptical and a credulous approach. However, defeat is restricted to rules with contradictory heads, making it difficult to represent more complex decisions. In [4], preference in extended disjunctive logic programming is considered. As far as overriding is concerned, the technique corresponds rather well with our skeptical defeating, but, again, alternatives are limited to an atom and its (classical) negation.

To reason about updates of generalized logic programs, extended logic programs without classical negation, [1] introduces dynamic logic programs. A stable model of such a dynamic logic program is a stable model of the generalized program obtained by removing the rejected rules. The definition of a rejected rule corresponds to our definition of a defeated rule when a and $\neg a$ are considered alternatives. A similar system is proposed in [11], where sequences are based on extended logic programs, and defeat is restricted to rules with opposing heads. The semantics is obtained by mapping to a single extended logic program containing expanded rules such that defeated rules become blocked in the interpretation of the "flattened" program. In [8], a mapping from extended logic programs to OCLP was presented. A very similar mapping allows us to map both dynamic logic programs as sequences of extended logic program to OCLP.

[2] added a system of preference to the dynamic logic programs of [1]. This preference is used to select the most preferred stable models. A similar mechanism is also used by [3] to obtain preferred answer sets: preferences are used to filter out unwanted candidate models, they are not used during model creation as is the case for OCLP.

[18] also proposes a formalism that uses the order among rules to induce an order on answer sets for inconsistent programs, making it unclear on how to represent decisions. Along the same line, [10] proposes logic programs with compiled preferences, where preferences may appear in any part of the rules. For the semantics, [10] maps the program to an extended logic program.

6 Conclusions and Directions for Future Research

In this paper we proposed a mechanism for transforming ordered choice logic programs to semi-negative logic program while preserving, depending on the transformation, the skeptical or credulous answer set semantics.

Having such a transformation allows an implementation of OCLP on top of answer set solvers like Smodels ([12]), and dlv ([17]). Clearly the mapping we proposed in this paper is more theoretical inspired and can be made more efficient by taking into account the various constructs provided by the answer set solvers. The disjunctive rules provided by dlv would reduce rules of type a), while the special choice construct of smodels would reduce both rules of type a) and d). Theoretically speaking, there is no need to introduce the atoms a_r. One can easily incorporate the bodies into the rules describing the defeating conditions. Unfortunately, this makes the mapping harder to read and would, in the credulous case, create more rules of type c). Writing the actual front-end, we need to investigate which option would be more efficient.

Previously, OCLP was used to describe and to reason about game theory ([8, 9]). To this extend, we used a special class of OCLPs. Each atom appears exactly once in a choice rule and non of the choice rules can be defeated. Combining this knowledge with the mapping of OCLP to logic programs, we can create a game-theory tailored front-end to answer set solvers.

In [9], we proposed a multi-agent system were the knowledge and beliefs of the agents is modeled by an OCLP. The agents communicate among each other by sending answer sets, skeptical or credulous, to each other. The notion of evolutionary fixpoint shows how the various agents reasoned in order to come to their final conclusions. Having an implementation for OCLP would allow us to implement multi-agent systems and run experiments in various domains. One possibility would be trying to incorporate this knowledge into Carel ([19]), a multi-agent system for organ and tissue exchange.

References

1. José Júlio Alferes, Leite J. A., Luís Moniz Pereira, Halina Przymusinska, and Teodor C. Przymusinski. Dynamic logic programming. In Cohn et al. [6], pages 98–111.
2. José Júlio Alferes and Luís Moniz Pereira. Updates plus preferences. In *European Workshop, JELIA 2000*, volume 1919 of *Lecture Notes in Artificial Intelligence*, pages 345–360, Malaga, Spain, September–October 2000. Springer Verslag.
3. Gerhard Brewka and Thomas Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356, April 1999.
4. Francesco Buccafurri, Wolfgang Faber, and Nicola Leone. Disjunctive Logic Programs with Inheritance. In Danny De Schreye, editor, *International Conference on Logic Programming (ICLP)*, pages 79–93, Las Cruces, New Mexico, USA, 1999. The MIT Press.
5. Francesco Buccafurri, Nicola Leone, and Pasquale Rullo. Disjunctive ordered logic: Semantics and expressiveness. In Cohn et al. [6], pages 418–431.
6. Anthony G. Cohn, Lenhard K. Schubert, and Stuart C. Shapiro, editors. *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning*, Trento, June 1998. Morgan Kaufmann.
7. Marina De Vos and Dirk Vermeir. On the Role of Negation in Choice Logic Programs. In Michael Gelfond, Nicola Leone, and Gerald Pfeifer, editors, *Logic Programming and Non-Monotonic Reasoning Conference (LPNMR'99)*, volume 1730 of *Lecture Notes in Artificial Intelligence*, pages 236–246, El Paso, Texas, USA, 1999. Springer Verslag.
8. Marina De Vos and Dirk Vermeir. Dynamic Decision Making in Logic Programming and Game Theory. In *AI2002: Advances in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, pages 36–47. Springer, December 2002.

9. 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, December 2002.
10. J. Delgrande, T. Schaub, and H. Tompits. Logic programs with compiled preferences. In W. Horn, editor, *European Conference on Artficial Intelligence*, pages 392–398, 2000.
11. Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits. On Properties of update Sequences Based on Causal Rejection. *Theory and Practice of Logic Programming*, 2(6), November 2002.
12. Thomas Eiter, Nicola Leone, Cristinel Mateis, Gerald Pfeifer, and Francesco Scarcello. The KR system `dlv`: Progress report, comparisons and benchmarks. In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *KR'98: Principles of Knowledge Representation and Reasoning*, pages 406–417. Morgan Kaufmann, San Francisco, California, 1998.
13. 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.
14. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of fifth logic programming symposium*, pages 1070–1080. MIT PRESS, 1988.
15. Els Laenens and Dirk Vermeir. A Universal Fixpoint Semantics for Ordered Logic. *Computers and Artificial Intelligence*, 19(3), 2000.
16. Vladimir Lifschitz. Answer set programming and plan generation. *Journal of Artificial Intelligence*, 138(1-2):39–54, 2002.
17. I. Niemelä and P. Simons. Smodels: An implementation of the stable model and well-founded semantics for normal LP. In Jürgen Dix, Ulrich Furbach, and Anil Nerode, editors, *Proceedings of the 4th International Conference on Logic Programing and Nonmonotonic Reasoning*, volume 1265 of *LNAI*, pages 420–429, Berlin, July 28–31 1997. Springer.
18. Davy Van Nieuwenborgh and Dirk Vermeir. Preferred answer sets for ordered logic programs. In *European Workshop, JELIA 2002*, volume 1919 of *Lecture Notes in Artificial Intelligence*, pages 432–443, Cosenza, Italy, September 2002. Springer Verlag.
19. Javier Vázquez-Salceda, Julian Padget, Ulises Cortés, Antonio López-Navidad, and Francisco Caballero. Formalizing an electronic institution for the distribution of human tissues. *Artificial Intelligence in Medicine*, 27(3):233–258, 2003. published by Elsevier.