IMPLEMENTING PRIORITIZED REASONING IN LOGIC PROGRAMMING

Luciano Caroprese, Irina Trubitsyna and Ester Zumpano
DEIS, University of Calabria, 87030 Rende, Italy

Keywords: Answer Set Optimization, Choice Optimization, Prioritized Logic Programming.

Abstract: Prioritized reasoning is an important extension of logic programming and is a powerful tool for expressing desiderata on the program solutions in order to establish the best ones. This paper discusses the implementation of the case of preference relation among atoms and introduces a system, called CHOPPER, realizing choice optimization recently proposed in (Caroprese et al., 2007). CHOPPER supports the $ASO_{\texttt{Ch}}$ and $ASO_{\texttt{FCh}}$ semantics based on the concept of “choice”, as a set of preference rules describing common choice options in different contexts, and the ASO semantics (Brewka et al., 2003), which valuates each preference rule separately. This paper outlines the architecture of the system, discusses aspects of the choice identification strategies and of the feasibility of choice options. Moreover, the comparison of the proposed approach with the other implementation approaches proposed in the literature is provided.

1 INTRODUCTION

Prioritized reasoning is an important extension of logic programming, used in a large variety of AI problems. The most common form of preference consists in specifying preference conditions among rules (Brewka and Eiter, 1999; Delgrande et al., 2000; Delgrande et al., 2003; Gelfond and Son, 1997), whereas, some other proposals admit the expression of preference relations among atoms (Brewka et al., 2003; Sakama and Inoue, 2000; Wakaki et al., 2003). See (Delgrande et al., 2004) for a survey on this topic.
This work is a contribution to realizing prioritized reasoning in logic programming in the presence of preference conditions involving atoms. This topic has been investigated in (Brewka et al., 2003) and (Sakama and Inoue, 2000), proposing the ASO and PLP semantics respectively. The ASO semantics evaluates the degree of satisfaction of all preference rules to determine the preferred models and can establish a preference relation between each couple of models directly. On the contrary, the PLP semantics compares two models only on the basis of their common preferences, and needs to test the transitive property to derive the non directly visible preference relations; this lies in a more complex implementation. A generalization of the ASO semantics which look beyond the each single preference rule, by considering preferences as a tool for choice representation, was recently proposed in (Caroprese et al., 2007).
The prioritized reasoning based on the choice optimization is illustrated by the following example.

Example 1 Consider the prioritized program $\langle \mathcal{P}_1, \Phi_1 \rangle$. $\mathcal{P}_1$ describes the possible menus by means of three rules r_1, r_2 and r_3, where $\oplus$ states that exactly one of the head's atoms has to be taken in the model, and three constraints c_1, c_2 and c_3, i.e. rules with empty heads satisfied if the body is false (e.g. c_1 states that $beef$ and red cannot be simultaneously present).

$$\begin{array}{ll}
\mathtt{r}_1: \mathtt{fish} \oplus \mathtt{beef} \leftarrow & \mathtt{c}_1: \leftarrow \mathtt{beef}, \mathtt{red} \\
\mathtt{r}_2: \mathtt{red} \oplus \mathtt{white} \leftarrow & \mathtt{c}_2: \leftarrow \mathtt{beef}, \mathtt{pie} \\
\mathtt{r}_3: \mathtt{pie} \oplus \mathtt{ice\text{-}cream} \leftarrow & \mathtt{c}_3: \leftarrow \mathtt{fish}, \mathtt{white} \\
\rho_1: \mathtt{white} > \mathtt{red} \leftarrow \mathtt{fish} & \\
\rho_2: \mathtt{red} > \mathtt{white} \leftarrow \mathtt{beef} & \\
\rho_3: \mathtt{pie} > \mathtt{ice\text{-}cream} \leftarrow &
\end{array}$$

$\mathcal{P}_1$ has three stable models $M_1 = \{\mathtt{fish}, \mathtt{red}, \mathtt{pie}\}$, $M_2 = \{\mathtt{fish},\ \mathtt{red},\ \mathtt{ice\text{-}cream}\}$ and $M_3 = \{\mathtt{beef}, \mathtt{white}, \mathtt{ice\text{-}cream}\}$. The preference rules ρ_1 and ρ_2 specify that (i) the choice of drink (*white* or *red*) follows the selection of the main dish ($fish$ or $beef$);

Caroprese L., Trubitsyna I. and Zumpano E. (2007).
IMPLEMENTING PRIORITIZED REASONING IN LOGIC PROGRAMMING.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - AIDSS*, pages 94-100
DOI: 10.5220/0002353800940100
Copyright © SciTePress

(ii) *white* is better than *red* in the presence of $fish$; (iii) *red* is better than *white* in the presence of $beef$; whereas ρ_3 denotes that *pie* is better than *ice-cream*. M_1 is better than M_2 as *pie* is preferred to *ice-cream* and they have the same choice option of drink. M_1 is better than M_3 as they have the second best choice of the drink (in the presence of $fish$ and $beef$ respectively), whereas the choice of the *dessert* is satisfied better by M_1. Thus M_1 is a preferred model.
Observe that the ASO semantics, which considers each preference rule separately, cannot establish the preference ordering between M_1 and M_3 and derives that both M_1 and M_3 are preferred models. In more details, it derives that M_1 *is better that* M_3 *w.r.t.* ρ_2 and M_3 *is better that* M_1 *w.r.t.* ρ_1, thus they cannot be compared. $\Box$

This paper discusses the implementation of choice optimization in logic programming and presents a system, called CHOPPER, realizing this kind of prioritized reasoning.

2 PRELIMINARIES

An *Answer Set Optimization* program, ASO, is a pair $\langle \mathcal{P}, \Phi \rangle$, where $\mathcal{P}$ is called *Generating Program* and Φ is called *Preference Program*; Φ consists of a finite set of rules of the form $C_1 > \cdots > C_k \leftarrow body$, where *body* is a conjunction of literals, i.e. atoms or negation of atoms, and C_is are *boolean combinations* of literals. Intuitively, a preference rule $\rho \in \Phi$ above described introduces a preference order among $C_1, ..., C_k$: C_i is preferred to C_j, for $i < j$ and $i, j \in [1..k]$. Thus, the set of preferences Φ determines a preference ordering on the answer sets described by $\mathcal{P}$.
Let $\Phi = \{r_1, ..., r_n\}$ be a preference program and S be an answer set, then S induces a *satisfaction vector* $V_s = [v_s(r_1), ..., v_s(r_n)]$ where: a) $v_s(r_j) = I$, if r_j is *Irrelevant* to S, i.e. (i) the body of r_j is not satisfied in S or (ii) the body of r_j is satisfied, but none of the C_is is satisfied in S. b) $v_s(r_j) = min\{i \mid S \models C_i\}$, otherwise. The satisfaction vectors are used to compare the answer sets under the assumption that I is equal to 1 (i.e., $v_{S_j}(r_i) = I$ is equivalent to $v_{S_j}(r_i) = 1$).
Let S_1 and S_2 be two answer sets, then (i) $S_1 \geq S_2$ if $V_{S_1} \leq V_{S_2}$, i.e. if $v_{S_1}(r_i) \leq v_{S_2}(r_i)$ for every $i \in [1..n]$; (ii) $S_1 > S_2$ if $V_{S_1} < V_{S_2}$, i.e. if $V_{S_1} \leq V_{S_2}$ and for some $i \in [1..n]$ $v_{S_1}(r_i) < v_{S_2}(r_i)$.
A set of literals S is an *preferred (optimal) model* of an ASO program $\langle \mathcal{P}, \Phi \rangle$ if S is an answer set of $\mathcal{P}$ and there is no answer set S' of $\mathcal{P}$ such then $S' > S$.
Given an ASO program $\langle \mathcal{P}, \Phi \rangle$, its computational complexity is one level above the complexity of the generating program $\mathcal{P}$ (Brewka et al., 2003).

ASO$_{\texttt{Ch}}$ Semantics. The generalization of ASO semantics, which captures the intuition depicted in the Example 1, is called the ASO$_{\texttt{Ch}}$ semantics. Differently from the ASO semantics, the ASO$_{\texttt{Ch}}$ semantics looks beyond the single preference rules, considering them as a tool for choice representation: head atoms correspond to the *choice options*, whereas the body of preference rule specifies the *choice context*, i.e. the decisions which have to precede this choice. The ASO$_{\texttt{Ch}}$ semantics admits only atoms or disjunction of atoms in the head of preference rules. If C_i is a disjunction of atoms, all atoms in C_i are equally good choice options; atoms in C_i are preferred to the atoms in C_j, for $i < j$ and $i, j \in [1..k]$. The evaluation strategy consists in the identification of choices and in the comparison of choices instead of single rules in order to select preferred models.
More specifically, the partition of Φ into a *set of choices* (subset of preference rules), denoted by $\texttt{Ch}(\Phi)$, is performed following the *choice identification strategy*: two preference rules ρ_1 and ρ_2 *define the same choice* $\texttt{Ch}$, denoted by $\rho_1, \rho_2 \in \texttt{Ch}$, if (i) ρ_1 and ρ_2 have at least one common atom in their heads; and (ii) $\exists \rho_3$ such that $\rho_1, \rho_3 \in \texttt{Ch}$ and $\rho_3, \rho_2 \in \texttt{Ch}$.
The ASO$_{\texttt{Ch}}$ semantics presumes that prioritized programs are *well-formed*, i.e. each set of preference rules $\rho_1, ..., \rho_k$ defining a choice is specified over alternative contexts. This hypothesis ensures that in each model M at most one of the different contexts of a choice is satisfied, i.e. $\forall$ model M, $\forall$ $\texttt{Ch}$, there is at most one preference rule $\rho \in \texttt{Ch}$, denoted as *active in M*, such that $body(\rho)$ is satisfied in M.
After constructing the set of choices $\texttt{Ch}(\Phi)$, the preferred stable models of $\langle \mathcal{P}, \Phi \rangle$ are computed by associating to each model M of $\mathcal{P}$ a satisfaction vector reporting the degree of satisfaction of each choice $\texttt{Ch} \in \Phi$. The evaluation strategy of ASO$_{\texttt{Ch}}$ semantics is illustrated by the following example.

Example 2 Consider the prioritized program $\langle \mathcal{P}_1, \Phi_1 \rangle$ reported in Example 1. The preference rules $\{\rho_1, \rho_2\} \in \Phi_1$, having common atoms in their heads, define the same choice, say $\texttt{Ch}_{\texttt{dr}}$, whereas the last preference rule ρ_3 defines another choice, say $\texttt{Ch}_{\texttt{d}}$. Consequently, the set of preference rules $\{\rho_1, \rho_2, \rho_3\} \in \Phi_1$ is partitioned into two different choices: $\texttt{Ch}_{\texttt{dr}} = \{\rho_1, \rho_2\}$ and $\texttt{Ch}_{\texttt{d}} = \{\rho_3\}$ that models the choice of drink and dessert respectively. The choice satisfaction vectors are $V_{\texttt{M}_1} = [2,1]$, $V_{\texttt{M}_2} = [2,2]$, $V_{\texttt{M}_3} = [2,2]$, consequently, M_1 is the preferred model owing to the dessert choice. Note that ASO semantics gives $V_{\texttt{M}_1} = [2, I, 1]$, $V_{\texttt{M}_2} = [2, I, 2]$, $V_{\texttt{M}_3} = [I, 2, 2]$ and returns M_1 and M_3 as preferred models. It does not admit the comparison of M_1 and M_3 owing to opposite satisfaction degrees of ρ_1 and ρ_2. $\Box$

Given a prioritized program $\langle \mathcal{P}, \Phi \rangle$, the ASO$_{\mathtt{Ch}}$ semantics coincides with the ASO semantics if $\nexists \rho_1, \rho_2 \in \Phi$ having at least one common atom in their heads.
The computational complexity of ASO$_{\mathtt{Ch}}$ semantics is not increased w.r.t. the ASO semantics. In fact, the choice identification can be done in polynomial time; while given a model M the evaluation of its choice satisfaction vector V_M can be performed by (i) identifying the active rule for each choice, which can be done in polynomial time; and (ii) establishing the satisfaction degrees of the active rules as in the case of ASO semantics.

ASO$_{\mathtt{FCh}}$ Semantics. Given a choice $\mathtt{Ch}$ and a model M, the ASO$_{\mathtt{Ch}}$ semantics evaluates M on the basis of the selected option in the active preference rule $\rho \in \mathtt{Ch}$. Observe that the head atoms of ρ describe the possible options of $\mathtt{Ch}$ and the preference among them, without taken into account their *feasibility*, i.e. the really possibility of selecting these options during the choice. This property depends on the constraints present in logic program and is determined by the *choice context* in a given model, i.e. by the set of atoms whose selection precede this choice. An alternative to the ASO$_{\mathtt{Ch}}$ semantics evaluating the choices on the basis on their really possible options is called ASO$_{\mathtt{FCh}}$ semantics.

Example 3 Consider the prioritized program $\langle \mathcal{P}_3, \Phi_3 \rangle$, describing different menus and the preferences among drinks and desserts:

$\mathtt{r}_1 : \mathtt{fish} \oplus \mathtt{beef} \leftarrow$ $\qquad$ $\mathtt{c}_1 : \leftarrow \mathtt{beef},\ \mathtt{pie}$
$\mathtt{r}_2 : \mathtt{red} \oplus \mathtt{white} \oplus \mathtt{beer} \leftarrow$ $\qquad$ $\mathtt{c}_2 : \leftarrow \mathtt{fish},\ \mathtt{ice\text{-}cream}$
$\mathtt{r}_3 : \mathtt{pie} \oplus \mathtt{ice\text{-}cream} \leftarrow$
$\rho_1 : \mathtt{white} > \mathtt{red} > \mathtt{beer} \leftarrow \mathtt{fish}$
$\rho_2 : \mathtt{red} \vee \mathtt{beer} > \mathtt{white} \leftarrow \mathtt{beef}$
$\rho_3 : \mathtt{pie} > \mathtt{ice\text{-}cream} \leftarrow \mathtt{beer}$

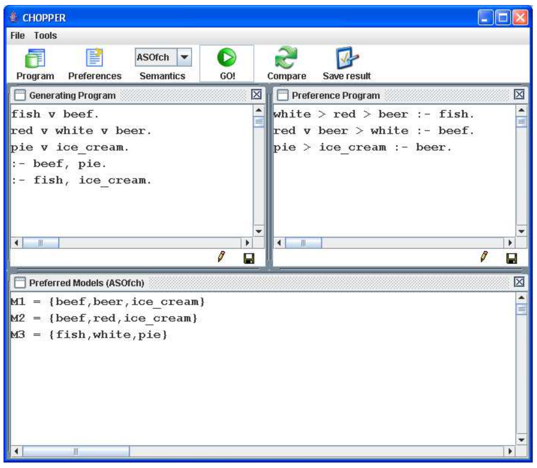
The generating program $\mathcal{P}_3$ has six stable models: $\mathtt{M}_1 = \{\mathtt{fish}, \mathtt{white}, \mathtt{pie}\}$, $\mathtt{M}_2 = \{\mathtt{fish}, \mathtt{red}, \mathtt{pie}\}$, $\mathtt{M}_3 = \{\mathtt{fish}, \mathtt{beer}, \mathtt{pie}\}$, $\mathtt{M}_4 = \{\mathtt{beef}, \mathtt{white}, \mathtt{ice\text{-}cream}\}$, $\mathtt{M}_5 = \{\mathtt{beef}, \mathtt{red}, \mathtt{ice\text{-}cream}\}$ and $\mathtt{M}_6 = \{\mathtt{beef}, \mathtt{beer}, \mathtt{ice\text{-}cream}\}$. Φ_3 defines the set of choices $\mathtt{Ch} = \{\mathtt{Ch}_{\mathtt{dr}}, \mathtt{Ch}_{\mathtt{d}}\}$, where $\mathtt{Ch}_{\mathtt{dr}} = \{\rho_1, \rho_2\}$ and $\mathtt{Ch}_{\mathtt{d}} = \{\rho_3\}$ describe the choices of drinks and desserts respectively.
In order to establish the set of preferred solutions the ASO$_{\mathtt{Ch}}$ semantics constructs the choice satisfaction vector for each model: $V_{\mathtt{M}_1} = [1, I\ ]$, $V_{\mathtt{M}_2} = [2, I\ ]$, $V_{\mathtt{M}_3} = [3, 1]$, $V_{\mathtt{M}_4} = [2, I\ ]$, $V_{\mathtt{M}_5} = [1, I\ ]$, $V_{\mathtt{M}_6} = [1, 2]$ and obtain M_1 and M_5 as a result. The ASO$_{\mathtt{FCh}}$ semantics takes into account the feasibility of choice options and substitutes $V_{\mathtt{M}_6} = [1, \mathbf{2}]$ by $V'_{\mathtt{M}_6} = [1, \mathbf{1}]$ following the intuition that its option of the dessert choice (*ice-cream*) is the unique (and, consequently, best) possible option. Thus M_6 is also a preferred model. $\square$

The implementation of the ASO$_{\mathtt{FCh}}$ semantics requires the test of the feasibility of choice options, which can be performed in a polynomial time. Thus, the computational complexity of ASO$_{\mathtt{FCh}}$ semantics is not increased w.r.t. the ASO$_{\mathtt{Ch}}$ semantics (Caroprese et al., 2007).

3 THE CHOPPER SYSTEM

CHOPPER (CHoice OPtimizer for PrioritizEd Reasoning) is an answer set optimization system realizing prioritized reasoning based on the choice evaluation.
The system prototype has been developed on top of the well-known DLV prover (Leone et al., 2002) by using Java 2 Platform. In particular, the answer set evaluation is performed with the DLV system, whereas the prioritized reasoning and the user interface are realized by means of personalized java procedures. We point out that besides DVL several other deductive systems based on stable model semantics have been proposed in the literature (Smodels, DeRes) (Cholewinski et al., 1996; Syrjanen and Niemela, 2001). The choice of DLV is orthogonal w.r.t. the development of the prototype, since CHOPPER can exploit any other deductive systems proposed in the literature.

Using of CHOPPER. CHOPPER receives in input the prioritized program and the specification of the semantics to be applied and extracts and returns the preferred stable models as a result. The system can be used by means of a *User Interface* - $\mathcal{UI}$ - which allows to specify (i) the prioritized program - $\langle \mathcal{P}, \Phi \rangle$, (ii) the semantics - $\mathcal{CS}$ - chosen among the semantics discussed in the paper and allows to visualize the obtained result, i.e. the set of preferred stable models - $\mathcal{PSM}$.
The running of CHOPPER for the program $\langle \mathcal{P}_3, \Phi_3 \rangle$ under the ASO$_{\mathtt{FCh}}$ semantics is provided in Figure 1.
In more details, the prioritized program is provided to the system by means of two different textual files, written following the DLV syntax, containing the generating program and the preference program, respectively. Note that in the preference program the system admits the symbol ">" for expressing the preference order among choice options.
CHOPPER also offers the possibility of comparing two different semantics by evidencing the set of models common to both the semantics. The result of comparison of ASO$_{\mathtt{Ch}}$ and ASO$_{\mathtt{FCh}}$ for the program $\langle \mathcal{P}_3, \Phi_3 \rangle$ is provided in Figure 2. The symbol "$*$"



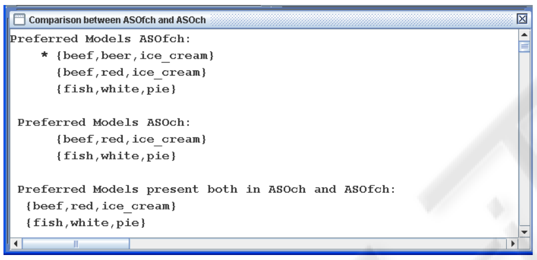
Generating Program
```
fish v beef.
red v white v beer.
pie v ice_cream.
:- beef, pie.
:- fish, ice_cream.
```

Preference Program
```
white > red > beer :- fish.
red v beer > white :- beef.
pie > ice_cream :- beer.
```

Preferred Models (ASOfch)
```
M1 = {beef,beer,ice_cream}
M2 = {beef,red,ice_cream}
M3 = {fish,white,pie}
```

Figure 1: The $\mathcal{UI}$ of CHOPPER.

marks the first model provided by the ASO$_{\mathtt{FCh}}$ semantics and states that this model is not provided by the ASO$_{\mathtt{Ch}}$ semantics.


Comparison between ASOfch and ASOch
```
Preferred Models ASOfch:
    * {beef,beer,ice_cream}
      {beef,red,ice_cream}
      {fish,white,pie}

 Preferred Models ASOch:
      {beef,red,ice_cream}
      {fish,white,pie}

 Preferred Models present both in ASOch and ASOfch:
  {beef,red,ice_cream}
  {fish,white,pie}
```

Figure 2: Comparison result.

CHOPPER Architecture. The overall architecture of the CHOPPER prototype is reported in Figure 3. The core of the system consists in two main blocks: the *Analyzer Block* and the *Evaluator Block* described below.

The *Analyzer Block* is constituted by two different modules: the $\mathcal{P} - analyzer$ and the $\Phi - analyzer$.

- The Φ-*analyzer* takes in input the set of preference rules Φ and the chosen semantics $\mathcal{CS}$, and computes $\mathtt{Ch}(\Phi)$, i.e. the partition of Φ into the set of choices by following the choice identification strategy.
- The $\mathcal{P}$-*analyzer* receives in input the generating program $\mathcal{P}$ and additional information gained from Φ-analyzer, and invokes the DLV prover (Leone et al., 2002) in order to obtain the stable models of the program $\mathcal{P}$, $\mathcal{SM}(\mathcal{P})$.
- As an additional task, in the case of the ASO$_{\mathtt{FCh}}$ semantics, the Φ-*analyzer* establishes the precedence relation among choices, while the $\mathcal{P}$-*analyzer* extracts the set of constraints of $\mathcal{P}$, $IC(\mathcal{P})$, necessary for testing the feasibility of choice options.

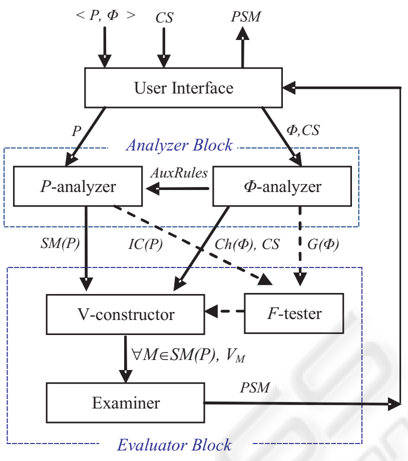


Figure 3: The architecture of CHOPPER.

The *Evaluator Block* is responsible for $\mathcal{PSM}$ discovering and is constituted by three modules: the *F-tester*, the *V-constructor* and the *Examiner*.

- The *F-tester* is able to test the feasibility of choice options.
- The *V-constructor* receives in input $\mathcal{SM}(\mathcal{P})$, $\mathcal{CS}$ and $\mathtt{Ch}(\Phi)$, and constructs the (feasible) choice satisfaction vector V_M for each stable model $M \in \mathcal{SM}(\mathcal{P})$.
- The *Examiner* module, performs the comparison of the choice satisfaction vectors so that establishing the set of preferred stable models $\mathcal{PSM}$. The result is provided to the $\mathcal{UI}$ responsible of user's communication.

4 IMPLEMENTATION

In the following we detail the implementation of the main modules constituting the *Analyzer Block* and the *Evaluator Block*.

4.1 Φ-analyzer

The Φ-analyzer examines the structure of preference rules in order to establish the partition of Φ into the set of choices $\mathtt{Ch}(\Phi)$ and to determine the order among choices. Moreover, it produces the set of auxiliary rules necessary to simplify the construction of the choice satisfaction vector without increasing the computational complexity of $\mathcal{P}$.

Choice identification. The choice identification process performed by the $\Phi-$*analyzer* depends on the selected semantics and is based on the choice identification strategy. In the case of ASO semantics each rule is associated with a different choice. In the case of $ASO_{\mathtt{Ch}}$ or $ASO_{\mathtt{FCh}}$ semantics the rules, having at least one common atom in their heads, are assigned to the same choice. The algorithm, performing this task, is presented in Figure 4. It takes in input a set of preference rules Φ and returns the partition of Φ into the set of choices $\mathtt{Ch}(\Phi)$.

```
Algorithm 1  Choice Identification
Input: A set of preference rules: $\Phi$;
Output: A set of choices: $\mathtt{Ch}(\Phi)$;
begin
   $\mathtt{Ch}(\Phi) = \{\}$;
   while ($\Phi$ is not empty) {
         $\rho = extractFirstElement(\Phi)$;
         $\mathtt{Ch} = \{\rho\}$; $\Phi = \Phi - \{\rho\}$;
         update=true;
         while (update==true ) {
               update=false;
               for each $\rho' \in \Phi$
                  for each $\rho \in \mathtt{Ch}$
                     if ($sameChoice(\rho, \rho')$) {
                        $\Phi = \Phi - \{\rho'\}; \mathtt{Ch} = \mathtt{Ch} + \{\rho'\}$;
                        $update = $ true; }
         }  $\mathtt{Ch}(\Phi) = \mathtt{Ch}(\Phi) + \mathtt{Ch}$;
   }  return $\mathtt{Ch}(\Phi)$;
end.
```

Figure 4: Choice Identification Algorithm.

At each step of the external iteration process the algorithm selects the first preference rule ρ from Φ and identifies and extracts all preference rules from Φ defining the same choice of ρ, called $\mathtt{Ch}$. The function *sameChoice* receives in input two preference rules ρ and ρ' and returns *true* if they have at least one common atom in their heads. In the positive case, ρ' is added to $\mathtt{Ch}$. The transitive property is supported by the nested *while*-block, which ensures that all preference rules defining the same choice are considered together. For the sake of simplicity of presentation, the optimization details, regarding the implementation of the transitive property are not reported in the above algorithm.

Choice ordering. The relationship among choices can be established by means of the *Choice dependency graph* described below.

Definition 1 *Choice dependency graph:* Given a set of choices $\Phi = \{\mathtt{Ch}_1, \ldots, \mathtt{Ch}_n\}$, its *dependency graph* $G(\Phi)$ is an oriented graph $\langle N, E\rangle$, where N denotes the set of nodes associated to choices, i.e $N = \{\mathtt{Ch}_1, \ldots, \mathtt{Ch}_n\}$, and E denotes the set of edges, $E = \{e(\mathtt{Ch}_i, \mathtt{Ch}_j) | \exists \rho_2 \in \mathtt{Ch}_j, \rho_1 \in \mathtt{Ch}_i$ and $\exists$ atom $\mathtt{A}$, *s.t.* $\mathtt{A} \in \mathtt{body}(\rho_2) \wedge \mathtt{A} \in \mathtt{head}(\rho_1)\}$. $\square$

The implementation of $G(\Phi)$ is really simple as its nodes correspond to the choices, whereas the presence of the edge $e(\mathtt{Ch}_i, \mathtt{Ch}_j)$ can be tested by considering the choice options of $\mathtt{Ch}_i$ and the positive body atoms, present in each $\rho \in \mathtt{Ch}_j$.

Definition 2 Given a set of choices $\Phi = \{\mathtt{Ch}_1, \ldots, \mathtt{Ch}_n\}$ and its *dependency graph* $G(\Phi)$, we say that the choice $\mathtt{Ch}_i$ *precedes* $\mathtt{Ch}_j$, denoted by $\mathtt{Ch}_i \prec \mathtt{Ch}_j$, if there is a path from the vertex $\mathtt{Ch}_i$ to the vertex $\mathtt{Ch}_j$, whereas there is no path from $\mathtt{Ch}_j$ to $\mathtt{Ch}_i$. $\square$

Previous definition allows the computation of the set of choices precedent to a choice $\mathtt{Ch}$: $\mathtt{Ch}'$ precedes $\mathtt{Ch}$ if there is a path from the vertex $\mathtt{Ch}'$ to the vertex $\mathtt{Ch}$; moreover, it takes into account the intuition that we cannot establish a precedence relation between two choices if they are involved in a cycle.

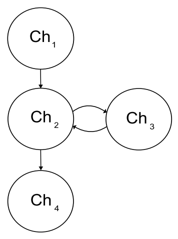


Figure 5: Choice Dependency Graph.

Example 4 Consider the Choice dependency graph presented in Figure 5. There are two nodes $\mathtt{Ch}_2$ and $\mathtt{Ch}_3$ involved in a cycle, thus we cannot establish the precedence relation between the corresponding choices. On the other hand, we can deduce, that $\mathtt{Ch}_1 \prec \mathtt{Ch}_2$, $\mathtt{Ch}_1 \prec \mathtt{Ch}_3$, $\mathtt{Ch}_1 \prec \mathtt{Ch}_4$, $\mathtt{Ch}_2 \prec \mathtt{Ch}_4$ and $\mathtt{Ch}_3 \prec \mathtt{Ch}_4$. $\square$

4.2 V-constructor and F-tester

The *V-constructor* and the *F-tester* modules collaborate in order to construct the choice satisfaction vector for each stable model of the generating program.

Choice Satisfaction Vector. The *V-constructor* receives in input $\mathcal{SM}(\mathcal{P})$, $\mathcal{CS}$ and $\mathtt{Ch}(\Phi)$, and constructs the choice satisfaction vector V_M for each sta-

ble model $M \in \mathcal{SM}(\mathcal{P})$ following the chosen semantics $\mathcal{CS}$. More specifically, given a model M, $V_M = [(\mathtt{v}(\mathtt{Ch_1}), \ldots, \mathtt{v}(\mathtt{Ch_n})]$ reports the satisfaction degree of each choice $\mathtt{Ch_i} \in \mathtt{Ch}(\Phi)$.
In the case of $\mathrm{ASO}_{\mathtt{Ch}}$ semantics each choice $\mathtt{Ch_i}$ is described by the set of preference rules, but at most one of them is active in M and $\mathtt{v}(\mathtt{Ch_i})$ reports its satisfaction degree. The cases than (i) no preference rule is active in M or (ii) there exists an active rule, but no choice option described by it is present in M, are considered as the best cases, i.e. $\mathtt{v}(\mathtt{Ch_i}) = \mathtt{1}$.
In the case of the ASO semantics each choice is described by a unique preference rule, thus V_M coincides with the satisfaction vector proposed in (Brewka et al., 2003).
If the $\mathrm{ASO}_{\mathtt{FCh}}$ is chosen, V_M has to be constructed by taking into account the feasibility of choice options. Consider, for instance, the choice $\mathtt{Ch_i}$ and suppose the choice option $\mathtt{d}$-th to be present in M, whereas $\mathtt{k}$ options preceding it are not feasible in M, then $\mathtt{v}(\mathtt{Ch_i}) = \mathtt{d} - \mathtt{k}$. The *V-Constructor* performs this task by interacting with the *F-tester* module, able to analyze the feasibility of choice options.

Feasibility Testing. The *F-tester* module is in charge of verifying the feasibility of the choice options present in a preference rules. The goodness of a model depends on the degree of satisfaction of a set of preference rules; anyhow, the measure of the degree of satisfaction of a preference rule, say ρ, active w.r.t. a model M should be evaluated by only considering the set of choice options feasible in M. In order to better depict this intuition, let's examine the program $\langle \mathcal{P}_3, \Phi_3 \rangle$ and the model M_6 reported in Example 3. ρ_3 is active in M_6 and as the choice of *pie* is not feasible in M_6, *ice-cream* becomes the best choice option as it is the only feasible one.
In order to capture this behavior the *F-tester* module computes the context of a choice $\mathtt{Ch}$ in a model M as the subset of atoms in M whose selection precedes $\mathtt{Ch}$. Intuitively, this subset of atoms can be individuated (i) by analyzing the choice dependency graph in order to establish the set of choices preceding $\mathtt{Ch}$, (ii) by identifying the atoms selected during these choices, or whose selection precede these choices.
More formally, given a set of choices $\mathtt{Ch}(\Phi)$, *the context of a choice* $\mathtt{Ch}$ w.r.t. a model M, denoted by $cntx_M(\mathtt{Ch})$, is the conjunction $\bigwedge_{\rho' \in \Omega(\mathtt{Ch}, M)} (body(\rho') \wedge best\_head(\rho')) \bigwedge body(\rho)$, where $\Omega(\mathtt{Ch}, M) = \{\rho' \mid \exists\, \mathtt{Ch}' \in \Phi \ s.t.\ \mathtt{Ch}' \prec \mathtt{Ch} \wedge \rho' \in \mathtt{Ch}' \wedge M \models body(\rho')\}$, $\rho \in \mathtt{Ch}$ is s.t. $M \models body(\rho)$ and $best\_head(\rho')$ is the best choice option of ρ' belonging to M.

Example 5 Consider the prioritized program $\langle \mathcal{P}_3, \Phi_3 \rangle$ from the Example 3. Notice that *beer* is the option of the drink choice and that each model M containing *beer* performs this choice following either the preference rule ρ_1 or ρ_2, i.e. before *beer*, M has selected either $fish$ or $beef$. Let's consider the model M_6: ρ_3 is active in M_6 as it contains *beer*, and *beer* follows the selection of $beef$. Thus the choice described by ρ_3 works in the presence (context) of $\{beef, beer\}$. $\square$

Given a model M and a choice $\mathtt{Ch}$, the module is therefore in charge of testing the feasibility of the options expressed in $\mathtt{Ch}$ by considering the constraints of the generating program and the context of $\mathtt{Ch}$ in M.

Definition 3 *Feasibility of choice options:* Given a prioritized program $\langle \mathcal{P}, \Phi \rangle$ defining the set of choices, a stable model M of $\mathcal{P}$, a choice $\mathtt{Ch} \in \mathtt{Ch}(\Phi)$ and a preference rule $\rho \in \mathtt{Ch}$, then $A_i \in head(\rho)$ is a *feasible* choice option w.r.t. M iff $A_i \wedge cntx_M(\mathtt{Ch}) \models \mathtt{IC}(\mathcal{P})$ and is *unfeasible* otherwise. $\square$

The test of the feasibility is performed by the *F-tester* module by constructing for each tested choice option a logic program which is satisfied if the tested option is feasible and fails otherwise. This program contains (i) the atoms of the choice context and a tested atom (choice option) as facts and (ii) the set of constraint $IC(\mathcal{P})$ of the generating program $\mathcal{P}$. The *F-tester* invokes the DLV prover in order to solve this program and if the program does not have stable models deduces that the tested option is unfeasible, or deduces that it is feasible otherwise.

5 OTHER APPROACHES

The computation of preferred models proposed in (Brewka et al., 2003) and (Wakaki et al., 2003) for ASO semantics and PLP semantics respectively is based on the use of a *tester program*[1]. In more details, given a prioritized logic program $\langle \mathcal{P}, \Phi \rangle$, the tester program takes in input a candidate solution S of $\mathcal{P}$ and looks for the other solutions of $\mathcal{P}$ (strictly) preferred w.r.t. S. This latter task is performed by generating the solutions of $\mathcal{P}$ and by comparing them with S.
The ASO semantics permits a direct comparison of two solutions, thus the tester program can be used as follows. The computation starts with an arbitrary solution S of $\mathcal{P}$. If the tester fails, S is an optimal solution and the computational process stops. Otherwise, a strictly better solution S_1 is discovered and the tester

[1]A similar technique was also implemented in (Janhunen et al., 2000; Brewka, 2002; Sakama and Inoue, 2000).

is run with S_1 as input. This process continues until an optimal solution is reached.
On the contrary, $\mathcal{PLP}$ semantics introduces the transitive property of preference relation, thus admits the preference relation between two solutions which are not directly comparable. Consequently, in this case two-step procedure is needed: in the first step all the direct preference relations among solutions have to be established; then the transitive relations can be discovered and the final conclusion can be derived. Wakaki et. al in (Wakaki et al., 2003) implement the direct comparisons by testing each answer set of $\mathcal{P}$ with a tester program; and then create an auxiliary logic program which extracts all preferred solutions on the basis of the preference relations generated at the previous step and those discovered by using the transitive property.

In the general case, both the approaches, previously described, need to perform more calls to the tester program which is in charge of computing the set of solution of $\mathcal{P}$. The approach adopted in CHOPPER aims to avoid the redundant computation of the set of solution of $\mathcal{P}$, performed during each call to the tester program. CHOPPER uses once the logic prover to find the set of solutions of the problem, and realizes the prioritized reasoning by means of personalized comparison procedures.

6 CONCLUSION

In this paper the implementation of prioritized reasoning in logic programming has been discussed. In particular, the case of preference relation among atoms has been investigated and a system, called CHOPPER, has been described. This system realizes choice optimization in logic programming by implementing the $\text{ASO}_{\texttt{Ch}}$ and $\text{ASO}_{\texttt{FCh}}$ semantics recently proposed in (Caroprese et al., 2007), and supports the ASO semantics (Brewka et al., 2003). In this paper the architecture of the system has been presented and aspects of the choice identification strategies and of the feasibility of choice options has been discussed. Moreover, the comparison of the proposed approach with the other implementation approaches proposed in the literature has been provided.

REFERENCES

Brewka, G. (2002). Logic programming with ordered disjunction. In *Proc. of the 18th Nat. Conf. on Artificial Intelligence (AAAI/IAAA)*, pages 100–105. AAAI Press.

Brewka, G. and Eiter, T. (1999). Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356.

Brewka, G., Niemela, I., and Truszczynski, M. (2003). Answer set optimization. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence*, pages 867–872. Morgan Kaufmann.

Caroprese, L., Trubitsyna, I., and Zumpano, E. (2007). A framework for prioritized reasoning based on the choice evaluation. In *Proc. of the 22nd Annual ACM Symposium on Applied Computing*.

Cholewinski, P., Marek, V. W., and Truszczynski, M. (1996). Default reasoning system deres. In *Proc. of the 5th Int. Conf. KR'97*, pages 518–528. Morgan Kaufmann.

Delgrande, J. P., Schaub, T., and Tompits, H. (2000). Logic programs with compiled preferences. In *Proc. of the 14th Eur. Conf. on Artificial Intelligence*, pages 464–468. IOS Press.

Delgrande, J. P., Schaub, T., and Tompits, H. (2003). A framework for compiling preferences in logic programs. *TPLP*, 3(2):129–187.

Delgrande, J. P., Schaub, T., Tompits, H., and K., W. (2004). A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence*, 20(2):308–334.

Gelfond, M. and Son, T. (1997). Reasoning with prioritized defaults. In *Proc. of the 3d Int. Workshop LPKR'97*, pages 164–223. Springer.

Janhunen, T., Niemela, I., Simons, P., and You, J.-H. (2000). Unfolding partiality and disjunctions in stable model semantics. In *Proc. of the 7th Int. Conf. KR'00*, pages 411–419. Morgan Kaufmann.

Leone, N., Pfeifer, G., Faber, W., Calimeri, F., Dell'Armi, T., Eiter, T., Gottlob, G., Ianni, G., Ielpa, G., Koch, K., Perri, S., and Polleres, A. (2002). The dlv system. In *Proc. of Eur. Conf. JELIA'02*, pages 537–540. Springer.

Sakama, C. and Inoue, K. (2000). Priorized logic programming and its application to commonsense reasoning. *Artificial Intelligence*, 123:185–222.

Syrjanen, T. and Niemela, I. (2001). The smodels system. In *Proc. of the 6th Int. Conf. LPNMR'01*, pages 434–438. Springer.

Wakaki, T., Inoue, K. nd Sakama, C., and Nitta, K. (2003). Computing preferred answer sets in answer set programming. In *Proc. of the 10th Int. Conf. LPAR'03*, pages 259–273. Springer.