Preferences and Priorities in ASP

S. Costantini  A. Formisano

Università di L’Aquila & Università di Perugia

CILC 2012 — Roma - June 2012
Answer set programming,... very briefly

- Syntax: an ASP program is a set of rules of the form
  \[ H \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m. \]

- Semantics: models are minimal, closed, supported set of atoms (literals)

- Common approach: represent a problem as a program whose models encode the solutions of the problem

- Applications in many fields (e.g., problem solving, planning, configuration,\ldots )
Answer Set Optimization

Introduced by [Brewka 2004], it refines the ASP framework and provides means to

- discriminate/compare/rank different answer sets, by
- modeling the different quality of different answer sets, by
- allowing the programmer to specify (qualitative or quantitative) preferences
Answer Set Optimization

Let \textit{prex} be a \textit{PDL expression}, namely, an expression that describes a (preference) order on sets of atoms (literals). Then, we have:
Let $prex$ be a PDL expression, namely, an expression that describes a (preference) order on sets of atoms (literals). Then, we have:

**Definition (Brewka 2004)**

An *answer set optimization problem* (AOP) is a pair

$$O = (P, prex)$$

where $P$ is a program and $prex$ a PDL expression. A solution of $O$ is an answer set of $P$ which is optimal according to the (pre)order represented by $prex$. 
Let \textit{prex} be a PDL expression, namely, an expression that describes a (preference) order on sets of atoms (literals). Then, we have:

\textbf{Definition (Brewka 2004)}

An \textit{answer set optimization problem} (AOP) is a pair

\[ O = (P, \text{prex}) \]

where \( P \) is a program and \textit{prex} a PDL expression. A solution of \( O \) is an answer set of \( P \) which is optimal according to the (pre)order represented by \textit{prex}.

How to specify \textit{prex}?
Preference rules

The simpler form of preference is described by preference rules:

**Definition**

Let $A$ be a set of atoms. A *preference rule* over $A$ is a writing of the form:

$$C_1:p_1 > \cdots > C_k:p_k \leftarrow B_1, \ldots, B_n, \text{not } D_1, \ldots, \text{not } D_m.$$  

where each $C_i$ is a Boolean condition (over $A$) and each $p_i$ is an integer.
Preferences and Priorities in ASP

Preference rules

The simpler form of preference is described by preference rules:

**Definition**

Let $A$ be a set of atoms. A *preference rule* over $A$ is a writing of the form:

$$C_1:p_1 > \cdots > C_k:p_k \leftarrow B_1, \ldots, B_n, \text{not } D_1, \ldots, \text{not } D_m.$$  

where each $C_i$ is a Boolean condition (over $A$) and each $p_i$ is an integer.

**Semantics:** whenever two sets $A_1, A_2$ satisfy the body of the rule, $A_1$ is preferred to $A_2$ if, for some $j$, 

$$A_1 \models C_j \quad \text{and} \quad j < \min \{ i \mid A_2 \models C_i \}$$
PDL expressions (syntax)

Simple PDL expressions are defined first:

**Definition**

The collection $PDL^p$ is defined as follows.

- if $r$ is a preference rule, then $r \in PDL^p$
- if $e_1, \ldots, e_k \in PDL^p$, then $(psum e_1, \ldots, e_k) \in PDL^p$
PDL expressions (syntax)

To combine basic preferences, general PDL expressions are then introduced:

**Definition**

The collection $PDL$ is defined as follows.

- if $r \in PDL^p$ then $r \in PDL$
- if $e_1, \ldots, e_k \in PDL^p$, then the expressions $(inc e_1, \ldots, e_k)$, $(rinc e_1, \ldots, e_k)$, $(card e_1, \ldots, e_k)$, and $(rcard e_1, \ldots, e_k)$ are in $PDL$
- if $e_1, \ldots, e_k \in PDL$, then $(pareto e_1, \ldots, e_k)$ and $(lex e_1, \ldots, e_k)$ are in $PDL$
PDL expressions (semantics)

- A order is induced on answer sets by introducing a notion of penalty
A order is induced on answer sets by introducing a notion of penalty

PDL expressions associate penalties to each answer set \( S \). For simple PDLs:

- If \( \text{prex} \) is a preference rule
  
  \[
  C_1 : p_1 > \cdots > C_k : p_k \leftarrow B_1, \ldots, B_n, \text{not } D_1, \ldots, \text{not } D_m.
  \]

  then, \( \text{pen}(S, \text{prex}) = p_j \) where \( j = \min\{i \mid S \models C_i\} \)
  (but, if \( S \) does not satisfy the body, let \( \text{pen}(S, \text{prex}) = 0 \))

- if \( \text{prex} \) has the form \( (psum \ e_1, \ldots, e_k) \), then
  \[
  \text{pen}(S, \text{prex}) = \sum_{i=1}^{k} \text{pen}(S, e_i)
  \]
PDL expressions (semantics)

Each PDL expression determines an order $\text{Ord}(\text{prex})$ on answer sets:
Given $S_1$ and $S_2$, and $\text{prex}$, we have $\text{Ord}(\text{prex})$, if one of the following holds:

- if $\text{prex}$ is a rule $r$, then $(S_1, S_2) \in \text{Ord}(\text{prex})$ (for short, $S_1 \geq S_2$) iff $\text{pen}(S_1, r) \leq \text{pen}(S_2, r)$

- if $\text{prex}$ has the form $(\text{pareto } e_1, \ldots, e_k)$, then $(S_1, S_2) \in \text{Ord}(\text{prex})$ iff $S_1 \geq_j S_2$, for each $j = 1, \ldots, k$, where $\geq_j$ is the order induced by $e_j$

- if $\text{prex}$ has the form $(\text{lex } e_1, \ldots, e_k)$, then $(S_1, S_2) \in \text{Ord}(\text{prex})$ iff $S_1 \geq_j S_2$, for all $j$ or $S_1 >_j S_2$, for some $j$ and for all $i < j$, $S_1 \geq_i S_2$

- etc...
Another form of preference in ASP

A form of preference reasoning has been introduced in RASP, an extension of ASP

- proposed to model and reason on processes that produce/consume resources
- in RASP, preferences may be conditional and context-dependent
- not necessarily described by a total/linear order (any binary relation can be used)
RASP-like preferences

**p-list:** Has the form $s_1 > \cdots > s_k$, and expresses a linear preference among the atoms $s_1, \ldots, s_k$

**cp-list:** a conditional p-list

\[ s_1 > \cdots > s_k \text{ pref \ when } L_1, \ldots, L_n, \]

is effective only when the condition $L_1, \ldots, L_n$ holds

**p-set:** has the form \( \{s_1, \ldots, s_k \mid \text{pred}\} \), where $s_1, \ldots, s_k$ are atoms and pred is a binary predicate that models a preference order among $s_1, \ldots, s_k$

P-lists, cp-lists, and p-sets can occur as heads or in the bodies of RASP-rules.
We need the auxiliary notion of derived rule:

**Definition**

Given a rule \( \gamma = B_0 \leftarrow B_1, \ldots, B_m \), the rule \( L_0 \leftarrow L_1, \ldots, L_m \) is derived from \( \gamma \) if, for each \( i \),

- \( L_i = B_i \) if \( B_i \) is a literal
- if \( B_i \) is a p-list \( s_1 > \cdots > s_k \),
  - or a cp-list \( (s_1 > \cdots > s_k \text{ pref}_\text{when} H_1, \ldots, H_n) \),
  - or a p-set \( \{s_1, \ldots, s_k \mid \text{pred}\} \), then
    - if \( i > 0 \), \( L_i = s_j \), for \( j \leq k \)
    - if \( i = 0 \), \( L_0 = s_j \) for \( j \leq k \), and moreover each \( \text{not } s_\ell \) is added to the body of the derived rule for \( \ell \neq j \).

Consequently, given a program, a derived program is defined as the collection of derived rules.
Extension of PDL’s syntax and semantics

Syntax:

- The collection PDL is extended including RASP-rules as members of $PDL^p$

- Given a program $\Pi$, involving RASP-like constructs, an extended answer set optimization problem is a pair $(\Pi', prex)$ where $\Pi'$ is the program derived from $\Pi$ and $prex$ is a PDL expression

Notice that, in evaluating $prex$, one must take into account the presence of p-lists, cp-lists, and p-sets in $\Pi'$
Extension of PDL’s syntax and semantics

Syntax:

- The collection PDL is extended including RASP-rules as members of $PDL^p$

- Given a program $\Pi$, involving RASP-like constructs, an extended answer set optimization problem is a pair $(\Pi', prex)$ where $\Pi'$ is the program derived from $\Pi$ and $prex$ is a PDL expression

Notice that, in evaluating $prex$, one must take into account the presence of p-lists, cp-lists, and p-sets in $\Pi'$,

how? ...
Extension of PDL’s syntax and semantics

Semantics:

- if $\text{prex}$ is a p-list $s_1 > \cdots > s_k$, then $\text{pen}(S, \text{prex}) = j$ where $j = \min\{i | S \models s_i\}$. And let $\text{pen}(S, \text{prex}) = \infty$ otherwise. (and similarly for cp-lists)

- if $\text{prex}$ is a p-set $\{s_1, \ldots, s_k | \text{pred}\}$, then $\text{pen}(S, \text{prex}) = j$ where $j = \min\{i | \forall j \leq k, j \neq i, S \models \text{pred}(s_i, s_j)\}$. And let $\text{pen}(S, \text{prex}) = \infty$ otherwise.
Extension of PDL’s syntax and semantics

Semantics:

- if `prex` is a p-list `s_1 \succ \cdots \succ s_k`, then `pen(S, prex) = j` where 
  \[ j = \min\{i \mid S \models s_i\} \]. And let `pen(S, prex) = \infty` otherwise. 
  (and similarly for cp-lists)

- if `prex` is a p-set \{`s_1, \ldots, s_k \mid pred`\}, then `pen(S, prex) = j` where 
  \[ j = \min\{i \mid \forall j \leq k, j \neq i, S \models pred(s_i, s_j)\} \]. 
  And let `pen(S, prex) = \infty` otherwise.

Finally,

- if `r` is a RASP-rule, we have `(S_1, S_2) \in Ord(r)` iff for each 
  p-list (cp-list, p-set) `s` occurring in `r`, it holds that 
  `pen(S_1, s) \leq pen(S_2, s)`. 
Future work: considering priority sequences?

- It is often the case that one justifies his/her preferences by means of reasons or criteria.
- Moreover, one might apply some kind of priority order on such criteria.
- Even more: both the priorities and the preferences may vary depending on the specific context at hand.
Future work: considering priority sequences?

- It is often the case that one justifies his/her preferences by means of reasons or criteria.
- Moreover, one might apply some kind of priority order on such criteria.
- Even more: both the priorities and the preferences may vary depending on the specific context at hand.

*Priority sequences* have been introduced (see [Liu 2009]) as a way to model these aspects. They have the general form:

$$C_1(X) \gg C_2(X) \gg \ldots \gg C_n(X)$$

where each $C_i$ represents a criterion.
Future work: PDL and priority sequences

Given a priority sequence $C_1(X) \gg C_2(X) \gg \ldots \gg C_n(X)$ a preference $\text{Pref}(a, b)$ between options $a$ and $b$ can be defined as:

$$\text{Pref}_1(a, b) = C_1(a) \land \neg C_1(b)$$
$$\text{Pref}_{k+1}(a, b) = \text{Pref}_k(a, b) \lor (\text{Eq}_k(a, b) \land C_{k+1}(a) \land \neg C_{k+1}(b))$$
$$\text{Pref}(a, b) = \text{Pref}_n(a, b)$$

(with $\text{Eq}_k(a, b) \equiv (C_1(a) \leftrightarrow C_1(b)) \land \cdots \land (C_k(a) \leftrightarrow C_k(b))$)
Future work: PDL and priority sequences

Given a priority sequence $C_1(X) \gg C_2(X) \gg \ldots \gg C_n(X)$ a preference $\text{Pref}(a, b)$ between options $a$ and $b$ can be defined as:

$$
\text{Pref}_1(a, b) = C_1(a) \land \neg C_1(b) \\
\text{Pref}_{k+1}(a, b) = \text{Pref}_k(a, b) \lor (\text{Eq}_k(a, b) \land C_{k+1}(a) \land \neg C_{k+1}(b)) \\
\text{Pref}(a, b) = \text{Pref}_n(a, b)
$$

(with $\text{Eq}_k(a, b) \equiv (C_1(a) \leftrightarrow C_1(b)) \land \cdots \land (C_k(a) \leftrightarrow C_k(b)))$

Proposal:

- Extend the framework by admitting priority sequences in rules
- Sequences are satisfied by a model $S$ with a degree of satisfaction, determined by which criteria $S$ satisfies
- Connection with PDL: higher degrees of satisfaction determine lower penalties
Preferences and Priorities in ASP