Answer Sets: From Constraint Programming Towards Qualitative Optimization

Gerhard Brewka

brewka@informatik.uni-leipzig.de

Universität Leipzig

Outline

1. Motivation
2. LPODs and optimization programs
3. Generic examples:
   - Abduction and diagnosis
   - Inconsistency handling
   - Solution coherence
4. A preference description language
5. Conclusions

The success of ASP

Main factors:

- availability of interesting implementations: dlv, Smodels, ASSAT ...
- shift of perspective from theorem proving to constraint programming/model generation
- many interesting applications in planning, reasoning about action, configuration, diagnosis, space shuttle control, ...

Natural next step: qualitative optimization brings in a lot of new interesting applications

Formalism I

LPOD: finite set of rules of the form:

$$C_1 \times \ldots \times C_n \leftarrow A_1, \ldots, A_m, \text{not } B_1, \ldots, \text{not } B_k$$

C_i, A_j, B_l ground literals.
if body then some C_j must be true, preferably C_1, if impossible then C_2, if impossible C_3, etc.

- Answer sets satisfy rules to different degrees.
- Use degrees to define global preference relation on answer sets.
- Different options how to do this (inclusion based, cardinality based etc.).

Formalism II

Optimization programs

- answer set generation independent of quality assessment
- P_{gen} generates answer sets, preference program P_{pref} compares them
- P_{pref} uses rules of the form

$$C_1 > \ldots > C_k \leftarrow body$$

C_i boolean combination built using $\vee$, $\wedge$, $\neg$, not .
$\neg$ in front of atoms, not in front of literals only.

Abduction and diagnosis

K program, H hypotheses, O observations

E explanation of O (dlv view) iff E minimal among

$$\{H' \subseteq H \mid S \in AS(H' \cup K), O \subseteq S, S \text{ consistent}\}$$

corresponding LPOD $P_{abd}(K, H, O)$:

$$\begin{aligned} K &\cup \{\leftarrow \text{not}\, o \mid o \in O\} \\ &\cup \{\neg ass(h) \times ass(h) \mid h \in H\} \\ &\cup \{h \leftarrow ass(h) \mid h \in H\}. \end{aligned}$$

E explanation iff S consistent answer set of $P_{abd}(K, H, O)$ and $E = \{h \in H \mid ass(h) \in S\}$

Consistency based diagnosis

- program P describes normal behavior using ab-predicates
- diagnosis minimal subset C' of components C such that

$$\{ab(c) \mid c \in C'\} \cup \{\neg ab(c) \mid c \in C \setminus C'\}$$

  explains observations O
- corresponding LPOD $P_{cd}(P, C, O)$:

$$P \cup \{\leftarrow \text{not}\, o \mid o \in O\} \cup \{\neg ab(c) \times ab(c) \mid c \in C\}$$

Inconsistency handling

- program P, possibly inconsistent; consistency restoring rules R
- names N_P and N_R for rules in P and R
- generate weakening of $P \cup R$ by replacing

$$head \leftarrow body \text{ with } head \leftarrow body, r_i$$

  where r_i rule's name
- add $\{r \times \neg r \mid r \in N_P\} \cup \{\neg r \times r \mid r \in N_R\}$
- minimal set of P-rules turned off, minimal set of R-rules turned on
- meta-preferences may express: P-rules to be neglected only if necessary

Solution coherence

- assume solution S for problem P was computed
- problem changes slightly to P'
- not interested in arbitrary solution of P', but solution *as close as possible to* S.
- distance measure based on symmetric difference: ($A \Delta B = A \setminus B \cup B \setminus A$)

$$S_1 \leq_S S_2 \textit{ iff } S_1 \Delta S \subseteq S_2 \Delta S$$

- corresponding preference program:

$$\{a > \mathrm{not}\, a \mid a \in S\} \cup \{\mathrm{not}\, a > a \mid a \notin S\}.$$

Meeting scheduling

$$
\begin{array}{lll}
part(p_1, m_1) & part(p_3, m_2) & unav(p_1, s_4) \\
part(p_2, m_1) & part(p_3, m_3) & unav(p_2, s_4) \\
part(p_2, m_2) & part(p_4, m_3) & unav(p_4, s_2)
\end{array}
$$

Meetings need 1 slot (using cardinality constraints):

$$1\{slot(M, S) : slot(S)\}1 \leftarrow meeting(M)$$

Constraints:

$$
\begin{array}{ll}
\leftarrow & part(P, M), slot(M, S), unav(P, S) \\
\leftarrow & part(P, M), part(P, M'), M \neq M', \\
 & slot(M, S), slot(M', S)
\end{array}
$$

Meeting scheduling, ctd.

A solution: $slot(m_1, s_1), slot(m_2, s_2), slot(m_3, s_3)$

p_4 becomes unavailable at s_3: $unav(p_4, s_3)$

Preference rules:
$slot(m_1, s_1) >$ not $slot(m_1, s_1)$,
$slot(m_2, s_2) >$ not $slot(m_2, s_2), \ldots$

Former solution invalid. Some new solutions:

$$S_1 : slot(m_1, s_1), slot(m_2, s_2), slot(m_3, s_4)$$
$$S_2 : slot(m_1, s_2), slot(m_2, s_1), slot(m_3, s_4)$$
$$S_3 : slot(m_1, s_3), slot(m_2, s_2), slot(m_3, s_1)$$

inclusion based strategy: S_1 better than S_2.
cardinality based strategy: S_1 better than S_2 and S_3.

Preference description language

- variety of existing preference combination strategies
- want to combine them in flexible ways
- PDL is a language for doing this
- consists of preference rules and (possibly nested) expressions

$$(comb\ e_1 \ldots e_n)$$

where $comb$ is a combination strategy, e_i an appropriate PDL expression.

Generalized preference rules

$$C_1{:}p_1 > \ldots > C_k{:}p_k \leftarrow body$$

C_i boolean combinations
p_i integer penalties satisfying $p_i < p_j$ whenever $i < j$.

$$C_1 > C_2 > \ldots > C_k \leftarrow body$$

abbreviates

$$C_1{:}0 > C_2{:}1 > \ldots > C_k{:}k\text{-}1 \leftarrow body$$

Syntax of PDL

PDL^p and PDL expressions:

1. r is preference rule $\Rightarrow r \in PDL^p$,
2. $e_1, \ldots, e_k \in PDL^p \Rightarrow (psum\ e_1 \ldots e_k) \in PDL^p$,
3. $e \in PDL^p \Rightarrow e \in PDL$,
4. $e_1, \ldots, e_k \in PDL^p \Rightarrow$ $(inc\ e_1 \ldots e_k)$, $(rinc\ e_1 \ldots e_k)$, $(card\ e_1 \ldots e_k)$ and $(rcard\ e_1 \ldots e_k) \in PDL$,
5. $e_1, \ldots, e_k \in PDL \Rightarrow$ $(pareto\ e_1 \ldots e_k)$ and $(lex\ e_1 \ldots e_k) \in PDL$.

Penalties and rule semantics

1. $prex = C_1{:}p_1 > \ldots > C_k{:}p_k \leftarrow body$
   S satisfies $body$ and at least one C_i:
   $pen(S, prex) = p_j$, where $j = min\{i \mid S \models C_i\}$,
   otherwise: $pen(S, prex) = 0$.
2. $prex = (psum\ e_1 \ldots e_k)$
   $pen(S, prex) = \sum_{i=1}^{k} pen(S, e_i)$.
3. $Ord(prex)$ preorder associated with $prex$, r rule:
   $(S_1, S_2) \in Ord(r)$ iff $pen(S_1, r) \leq pen(S_2, r)$.

Complex expressions

$\geq_i$ preorder ($>_i$ partial order) represented by e_i,
i, j range over $\{1, \ldots, k\}$, $P_S^p = \{j \mid pen(S, e_j) = p\}$

- $(S_1, S_2) \in Ord(pareto\ e_1 \ldots e_k)$ iff
  $S_1 \geq_j S_2$ for all j.
- $(S_1, S_2) \in Ord(lex\ e_1 \ldots e_k)$ iff
  $S_1 \geq_j S_2$ for all j or
  $S_1 >_j S_2$ for some j, and for all $i < j$: $S_1 \geq_i S_2$.
- $(S_1, S_2) \in Ord(inc\ e_1 \ldots e_k)$ iff $P_{S_1}^0 \supseteq P_{S_2}^0$.
- $(S_1, S_2) \in Ord(rinc\ e_1 \ldots e_k)$ iff
  $pen(S_1, e_j) = pen(S_2, e_j)$ for all j or
  $P_{S_1}^p \supset P_{S_2}^p$ for some p and $P_{S_1}^q = P_{S_2}^q$ for $q < p$.

Complex expressions, ctd.

- $(S_1, S_2) \in Ord(card\ e_1 \ldots e_k)$ iff $|P^0_{S_1}| \geq |P^0_{S_2}|$.
- $(S_1, S_2) \in Ord(rcard\ e_1 \ldots e_k)$ iff $|P^p_{S_1}| = |P^p_{S_2}|$ for all p or $|P^p_{S_1}| > |P^p_{S_2}|$ for some p, and $|P^q_{S_1}| = |P^q_{S_2}|$ for all $q < p$.
- $(S_1, S_2) \in Ord(psum\ e_1 \ldots e_k)$ iff $\sum_{i=1}^{k} pen(S_1, o_i) \leq \sum_{i=1}^{k} pen(S_2, o_i)$.

Special cases

1. preference progs $\{r_1, \ldots, r_k\}$: $(pareto\; r_1 \ldots r_k)$
2. ranked preference progs:
   $(lex\; (pareto\; r_{1,1} \ldots r_{1,k_1})...(pareto\; r_{n,1} \ldots r_{n,k_n}))$
3. cardinality and inclusion based combinations:
   use $rinc$ and $rcard$
4. weak constraints:
   $\leftarrow body.\; [w]$: use $\top{:}w \leftarrow body$ with $psum$
   $\leftarrow body.\; [w{:}l]$: group wrt. priority level l:
   $(lex\; (psum\; r_{1,1} \ldots r_{1,k_1})...(psum\; r_{n,1} \ldots r_{n,k_n}))$
5. $minimize\{a_1 = w_1, \ldots, a_k = w_k\}$ statements:
   single statement: $(psum\; a_1{:}w_1 \ldots a_k{:}w_k)$
   sequence: $(lex(psum \ldots)...(psum \ldots))$

Tester programs

- $T(P, M, prex)$ based on generating program P, current answer set M, compilation of $prex$
- generates answer sets strictly better than M
- generate and improve optimization strategy
- compilation example $(lex\ e_1 \ldots e_k)$:

$$geq_i \leftarrow geq_{i.1}, \ldots\ , geq_{i.k}$$
$$geq_i \leftarrow better_i$$
$$better_i \leftarrow better_{i.1}$$
$$better_i \leftarrow geq_{1.1}, better_{i.2}$$
...
$$better_i \leftarrow geq_{i.1}, \ldots geq_{i.k-1}, better_{i.k}$$

Conclusion

- ASP: successful declarative problem solving paradigm
- optimization facilities greatly increase applicability
- context dependent preferences among formulas flexible and powerful
- applications in diagnosis, planning, inconsistency, configuration with weak constraints, ...
- foundations of a preference description language for specifying flexible optimization strategies