WASP

Applications and Proofs-of-Concept

Dirk Vermeir (VUB)

Helsinki University of Technology, HUT
Technische Universität Wien, TUW
University of Bath, BATH
University of Cyprus, UCY
University of Manchester, UMAN
Vrije Universiteit Brussel, VUB

Application Areas

- **Planning**
- Web
- **Verification and Configuration**
- Multi-agent systems
- Security and Cryptanalysis
- **Diagnostic systems**
- Game theory
- ...

Preliminaries

Logic Programs

- A literal is an atom a or a negated atom $\neg a$.
- An extended literal is a literal or of the form $not\ l$ where l is a literal (negation as failure).
- For α a set of (extended) literals, $\alpha^- = \{a | not\ a \in \alpha\}$.
- An extended logic program (ELP) is a countable set of rules $\alpha \leftarrow \beta$, with $\alpha \cup \beta$ a finite set of extended literals.
- An interpretation I is a consistent set of ordinary literals.
- For an ordinary literal a, $I \models a$ iff $a \in I$, $I \models not\ a$ iff $a \notin I$. For a rule $r : \alpha \leftarrow \beta$, $I \models r$ if $\exists l \in \alpha \cdot I \models l$ whenever $I \models \beta$.

Answer Sets for programs without not

- If P is not -free, i.e. it contains only ordinary literals, an answer set is a (subset) minimal interpretation M such that $\forall r \in P \cdot M \models r$.
- Example.

$$p \vee q \leftarrow$$
$$\neg r \leftarrow p$$

Has answer sets $\{p, \neg r\}$ and $\{q\}$.

Answer sets for general programs

- For a general program P and an interpretation I, define the Gelfond-Lifschitz transformation

$$P^I = \{\alpha \backslash \alpha^- \leftarrow \beta \backslash \beta^- \mid I \models \alpha^- \ \wedge \ I \models \mathit{not}\ \beta^-\}$$

  P^I is free of not -literals.

- M is an answer set of P if it is an answer set of P^M.
- Example: let $P = \{a \leftarrow \mathit{not}\ b.\ \ b \leftarrow \mathit{not}\ a.\}$. Then $P^{\{a\}} = \{a \leftarrow\}$ and thus $\{a\}$ is an answer set (and so is $\{b\}$).
- Answer sets are not (anymore) necessarily subset-minimal: $p \vee \mathit{not}\ p \leftarrow$ has two answer sets $\emptyset$ and $\{p\}$.

Planning

Planning with ASP

Intuition:

- Time is discreet, bounded.
- Usual taxonomy of predicates: actions, fluents, background.
- Actions may have complex pre- and postconditions.
- Nondeterminism about (non) execution of an action is modelled by rules of the form.

$$action(T) \vee \neg action(T) \leftarrow precondition.$$

- Goals are modelled as constraints: $goal \leftarrow conditions(T)$ and $\leftarrow not\ goal$.

Planning with ASP: Example

- Four persons need to cross a bridge at night.
- The bridge can hold at most 2 persons.
- You cannot cross without a lamp.

? Plan to accomplish this task.

Example in DLV$^{\mathcal{K}}$

- actions:

```
cross2(X,Y) requires person(X), person(Y), X!=Y.
cross(X) requires person(X).
takelamp(X) requires person(X).
```

- fluents (and their types):

```
across(X) requires person(X).
diffside(X,Y) requires person(X), person(Y).
haslamp(X) requires person(X).
```

- initially:

```
caused -across(X). haslamp(a).
```

Example in DLV$^{\mathcal{K}}$ (cont'd)

- always:

```
executable cross2(X,Y) if haslamp(X).
executable cross2(X,Y) if haslamp(Y).
nonexecutable cross2(X,Y) if diffsides(X,Y).
..
caused across(X) after cross2(X,Y), -across(X).
caused -across(X) after cross2(X,Y), across(X).
..
caused across(X) if not -across(X)
  after across(X). % inertia
```

- goal:

```
across(a). across(b). across(c). across(d). (i)
```

Translation to ASP

- Represent time as a discrete sequence $time(1), \ldots time(N), next(1,2), \ldots next(N-1, N)$.
- For each causation rule "`caused H if B after A`", add (roughly, if `A` not empty) rule:

$$H(T_2) \leftarrow B(T_2), A(T_1), next(T_1, T_2), (requiredstuff)$$

- For each "`executable A if B` " condition, add a choice rule:

$$A(T_1) \vee \neg A(T_1) \leftarrow B(T_1), next(T_1, T_2).$$

- Initial conditions hold at time 0: $C(0)$.
- The goal G must be reached in i steps: $ok : -G(i)$. and $: -not\ ok$.

Minimal cost plans

- Performing an action may have a cost which may depend on the circumstances.
- In DLV$^{\mathcal{K}^c}$:

```
cross2(X,Y) requires .. costs S
  where speed(X,Sx), speed(Y,Sy),
    max(Sx, Sy, S).
```

- Translation:

$$cost_a(T, C) \leftarrow a(T), costconditions(C), U = T + 1$$

and the weak constraint $\Leftarrow cost_a(T, S)\ [S :]$ (best answer sets have minimal total cost of violated weak constraints).

Web

ASP applications and the Semantic Web

- Provide advanced reasoning services in the context of the semantic web.
- Need for declarative methods that can deal with default and preference information (several ASP approaches/implementations available).
- Ontologies:
  - Updates of knowledge bases.
  - ASP with infinite models as alternative for DLs
- Links with several other EU initiatives (e.g. INFOMIX, REWERSE, . . .).

Verification and Configuration

Verification using Smodels

Smodels supports some convenient extensions, e.g. choice rules

$$n\{p_1, \ldots, p_k\}m \leftarrow \ldots$$

select between n and m literals from the head, while

$$\leftarrow n\{q_1, \ldots, q_k\}$$

restrict the model to contain less than n atoms from $\{q_1, \ldots, q_k\}$.

Bounded Reachability using Smodels

- For 1-safe P/T Petri nets $\langle P, T, F\rangle$ (or LTL).
- Is a state satisfying C reachable in n steps?
- Discreet time $0, \ldots n$.
- Transitions rules:

$$\begin{array}{rcl} \{t(i)\} & \leftarrow & p_1(i)\ldots p_l(i).\textit{\% fire or not} \\ p(i+1) & \leftarrow & t(i) \\ & \leftarrow & 2\{t_p^1(i), \ldots, t_p^k(i)\}.\textit{\%}\ t_p^1, \ldots, t_p^k\ \textit{share}\ p \end{array}$$

- Frame axioms:

$$p(i+1) \leftarrow p(i), not\ t_p^1(i), \ldots, not\ t_p^k(i)$$

- Target constraint: $\leftarrow not\ C$.

Configuration using smodels

- Models:
  - Choice of optional components.
  - Required components (depending on configuration).
  - Incompatibilities.
  - Defaults.
- Example:

$$
\begin{array}{rcl}
computer & \leftarrow & \\
IDEdisk|SCSIdisk|floppydrive & \leftarrow & computer \\
FinnishKB|UKKB & \leftarrow & computer \\
 & \leftarrow & FinnishKB, UKKB \\
SCSIcontroller & \leftarrow & SCSIdisk
\end{array}
$$

Configuration using LPOD

- LPOD: ordered disjunction $a \times b \times c \leftarrow d$: if d then prefer a, else b, else c.
- Preferred answer set semantics attempts to maximally satisfy preferred alternatives (existence of preferred answer set containing a is Σ_2^P-complete).
- Configuration preferences:

$$
\begin{array}{rcl}
\mathit{emacs21.1} \times \mathit{emacs19.34} & \leftarrow & \mathit{emacs} \\
\mathit{libc6} \times \mathit{libc6dev} & \leftarrow & \mathit{needlibc6}, \mathit{not}\ \mathit{developer} \\
\mathit{libc6dev} \times \mathit{libc6} & \leftarrow & \mathit{needlibc6}, \mathit{developer}
\end{array}
$$

Multi-agent systems

Multi-agent systems

- Agents are represented by logic programs. Agents/programs communicate via uni-directional channels that transfer answer sets. Stable configurations represent consensus.
- ASP has been integrated into the declarative DALI language for representing multi-agent systems.

Security and Cryptanalysis

Security and Cryptanalysis

- Specification and verification of security protocols using reasoning about actions in an ASP language (smodels).
- Encodings of DES as ASP programs are competitive with SAT-oriented encodings.
- Open logic programs for policy verification.

Diagnosis

Diagnosis using Ordered Logic

- An ordered program is a partially ordered set of (named) deterministic rules $\langle R, <\rangle$: no disjunction, no not . Intuitively, $r_1 < r_2$ if (satisfaction of) r_1 is preferred over r_2.
- An answer set of a set of rules R may fail to satisfy (defeat) a rule $a \leftarrow \alpha$ provided that it applies a competing rule $\neg a \leftarrow \beta$.
- The reduct of $\langle R, <\rangle$ w.r.t. an interpretation I consists of the set of satisfied rules $\{r \mid I \models r\}$.
- For reducts $R_1 \sqsubseteq R_2$ iff $\forall r_2 \in R_2 \backslash R_1 \cdot \exists r_1 \in R_1 \backslash R_2 \cdot r_1 < r_2$.
- Preferred answer sets have $\sqsubset$-minimal reducts.

Ordered Logic

- Ordered logic has similar complexity as DLP (Σ_2^P for deciding whether there exists a preferred answer set containing a).
- Adding not does not increase the expressiveness.
- Example (simulation of traditional LP):

$$\frac{\begin{array}{c}\neg a \leftarrow \\ \neg b \leftarrow\end{array}}{\begin{array}{c}a \leftarrow \neg b \\ b \leftarrow \neg a\end{array}}$$

has $\{a, \neg b\}$ and $\{\neg a, b\}$ as preferred answer sets.

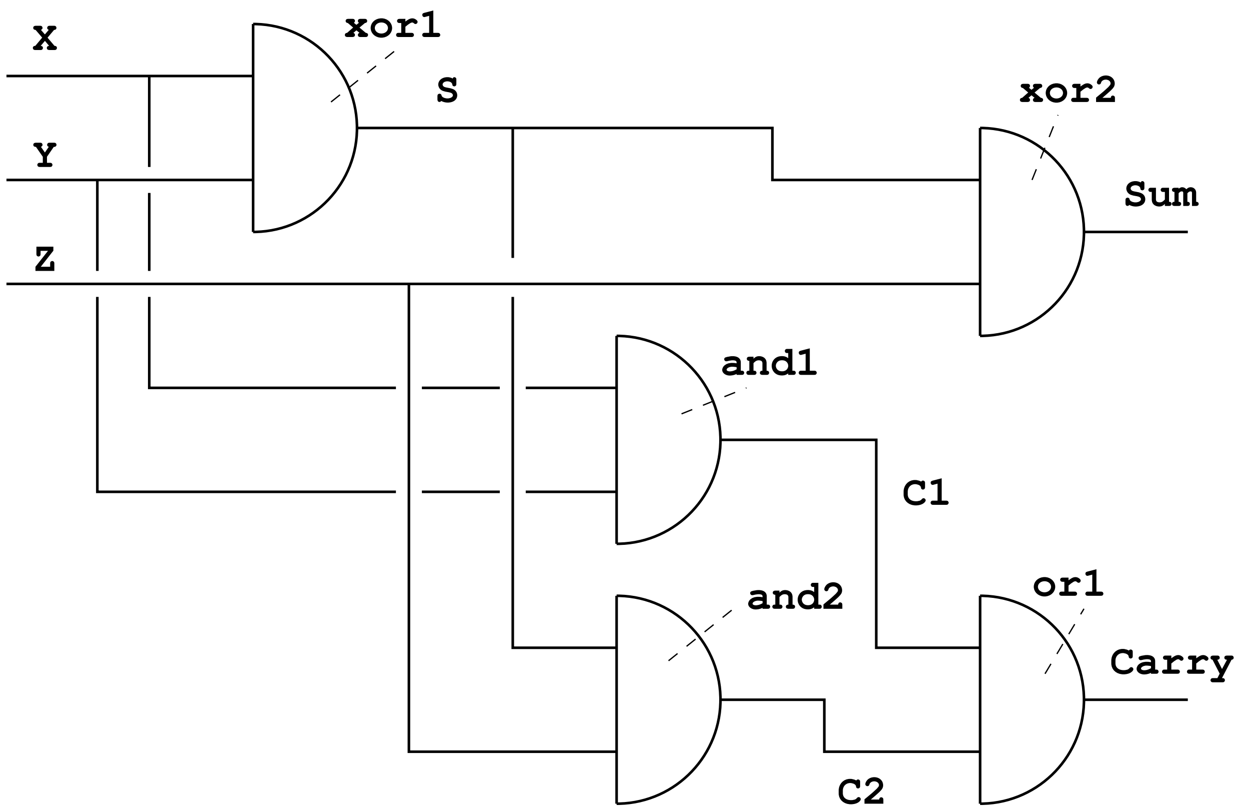
Diagnosis with OLP

- Intuition: organize the rules such that

  $\boxed{\textit{observations} < \textit{system description} < \textit{fault model}}$

- Observations are encoded using constraints of the form $\leftarrow \neg o$, where o is the observation.
- If the observations fit the normal description, the reduct of the preferred model will contain all observations and the system description.
- If the observations contradict the system description, the semantics will defeat some description rules and satisfy some fault model rules, in order to keep the observations in the answer set (“explain them”).

Example: binary adder



- A gate may be stuck at 1 ($s1$) or 0 ($s0$)
- (example from [flack94])

Example: main model

```
Model {
  adder(X,Y,Z, Sum, Carry) :- xor(xor1, X,Y,S),
    xor(xor2, Z,S,Sum), and(and1, X,Y,C1),
    and(and2, Z,S,C2), or(or1,C1,C2,Carry).
  xor(N, 1,1,0) :- port(N).
  and(N, 1,1,1) :- port(N).
  or(N, 1,1,1) :- port(N).
  ..
  % behaviour of broken gates
  xor(N, 0, 0, 1) :- port(N), fault(N, s1).
  xor(N, 0, 1, 0) :- port(N), fault(N, s0).
  ..
 }
```

Example (cont'd)

- Fault model, observations.

```
Error { fault(N, F). }
Default { % simulates naf
  -fault(N, F) :- port(N).
  -adder(X,Y,Z, Sum, Carry).
  }
Observations { :- -adder(0,0,1,0,1).  }
Model < Default < Error
```

- Preferred answer sets are minimal explanations: $\{fault(xor1, s1)\}$, $\{fault(or1, s1), fault(xor2, s0)\}$, $\{fault(and2, s1), fault(xor2, s0)\}$, $\{fault(and1, s1), fault(xor2, s0)\}$.
- Prototype implementation available.

Game Theory

Game Theory

OCLP, a variant of ASP, can be used to obtain a natural representation of finite extensive games with perfect information. Depending on the encoding, answer sets correspond with the Nash or subgame-perfect equilibria.

Role of VUB

- Researchers (PhD students): Stijn Heymans, Davy Van Nieuwenborgh.
- WP3 Extensions
  - Ordered Logic Programs
  - ASP with infinite models (but still decidable)
- WP5 Applications
  - OLPS implementation
  - Abduction and applications
  - Ontology language based on ASP with infinite models
- Results have been/are to be published in several papers in refereed international conferences and journals (10 so far).