Integrating Answer Set Programming and Satisfiability Modulo Theories

Ilkka Niemelä

Department of Information and Computer Science
Aalto University
Ilkka.Niemela@tkk.fi

Answer Set Programming (ASP)

- Term coined by Vladimir Lifschitz
- An approach to modeling and solving knowledge intensive search problems with defaults, exceptions, constraints, . . . : planning, configuration, model checking, network management, linguistics, combinatorics, . . .
- Solving a problem in ASP:
  Encode the problem as a logic program such that **solutions** to the problem are given by stable models (**answer sets**) of the program.

Problem $\longrightarrow$	ENCODING	Program $\longrightarrow$	ASP solver	Models $\longrightarrow$ (Solutions)
Data $\longrightarrow$	ENCODING			

Example: Hamiltonian cycles

A Hamiltonian cycle: a closed path that visits all vertices of the graph exactly once.

```
% Data
vtx(a). ... edge(a,b). ...
init_vtx(a0).   % for some vertex a0
% Problem encoding
{ hc(X,Y) } :- edge(X,Y).
:- 2 { hc(X,Y):edge(X,Y) }, vtx(X).
:- 2 { hc(X,Y):edge(X,Y) }, vtx(Y).
:- vtx(X), not r(X).
r(Y) :- hc(X,Y), edge(X,Y), init_vtx(X).
r(Y) :- r(X), hc(X,Y), edge(X,Y).
```


ASP—cont'd

- ASP solvers need to handle two challenging tasks: complex data and search
- Current systems employ a two level architecture with two steps:
- **Grounding** step handles complex data:
  - Given program *P* with variables, generate a set of ground instances of the rules preserving stable models.
  - LP and DDB techniques employed
- **Model search** for ground programs

☞ **ASP = KR + DDB + search**

Integrating ASP and SMT

- Solvers for the **propositional satisfiability problem (SAT)** are used widely as platforms for solving the model search problem.
- Interesting extensions of SAT studied recently: **Satisfiability Modulo Theories (SMT)**
- Efficient SMT solvers for expressive theories (integers, reals, uninterpreted function with equality, bit vectors, arrays, ...) are becoming available (`http://www.smtcomp.org/`)
- Is it possible to integrate ASP and SMT to exploit the strengths of both approaches?

Integrating ASP and SMT

- Interesting previous work on combining ASP and CSP techniques based on using an ASP and CSP solver together, for example,
  (El-Khatib, Pontelli, & Son, 2004; Baselice, Bonatti, & Gelfond 2005; Mellarkod & Gelfond 2007; Mellarkod, Gelfond & Zhang 2008)
- Here we study how ASP and SMT solver technology could be integrated.
  - We show how ground LPs with the stable model semantics can be embedded succinctly to a simple extension of SAT called difference logic supported by most SMT solvers.
  - Based on the embedding we demonstrate how to extend an ASP language with expressive constraints in such a way that an efficient implementation of the language can be obtained using off-the-shelf SMT solver technology.

Outline

- Stable models and propositional satisfiability
- Stable models and linear constraints
- Satisfiability Modulo Theories
- Translating LPs to SMT
- Integrating ASP and SMT

Preliminaries

- For propositional (ground) normal rules r of the form

$$a \leftarrow b_1, \ldots, b_m, \text{not } c_1, \ldots, \text{not } c_n.$$

  where $\mathrm{H}(r) = a$, $\mathrm{B}(r) = \{b_1, \ldots, b_m, \text{not } c_1, \ldots, \text{not } c_n\}$, $\mathrm{B}^+(r) = \{b_1, \ldots, b_m\}$, $\mathrm{B}^-(r) = \{c_1, \ldots, c_n\}$ the stable model semantics is defined as follows:
- A set of atoms M is a **stable model** of a program P iff M is the unique minimal set of atoms satisfying the reduct P^M, i.e., $M = LM(P^M)$ where

$$P^M = \{\mathrm{H}(r) \leftarrow \mathrm{B}^+(r) \mid r \in P, \mathrm{B}^-(r) \cap M = \emptyset\}.$$

- For a set of rules with variables stable models are defined to be those of the Herbrand instantiation of the rules.

Stable Models and SAT

- LPs with stable models are closely related to SAT through program completion.
  Example. P : Completion $\mathrm{CC}(P)$:
  $a \leftarrow b, \text{not } c$ $\quad (a \leftrightarrow ((b \wedge \neg c) \vee (\neg b \wedge d)))\wedge$
  $a \leftarrow \text{not } b, d$ $\quad \neg b \wedge \neg c \wedge \neg d$
- **Supported models** of a program and **models** of its completion coincide (Marek & Subrahmanian 1992)
- For **tight programs** (no positive recursion) **supported and stable models** coincide (Fages 1994).
- SAT solvers provide an interesting platform for implementing ASP solvers.

Stable Models and SAT

- However, translating general (non-tight) LPs to SAT is challenging
  - Modular translation not possible (I.N. 1999)
  - Without new atoms exponential blow-up (Lifschitz & Razborov 2006)
- There are one pass translations:
  - Polynomial size (Ben-Eliyahu & Dechter 1994; Lin & Zhao 2003)
  - $O(\|P\| \times \log |At(P)|)$ size (Janhunen 2004)
- Also incremental translations have been developed extending the completion dynamically with **loop formulas** (Lin & Zhao 2002)
  ☞ ASSAT and CMODELS ASP solvers

Stable Models and SAT

- Question: what needs to be added to SAT to allow a compact linear size translation of LPs to SAT?
- A possibility: stable models can be characterized using **orderings** (Elkan 1990; Fages 1994).
- Such an ordering can be captured with a restricted set of linear constraints on integers using **level rankings** (I.N. AMAI 2008)
- A suitable simple extension of propositional logic with such restricted linear constraints called **difference logic** is supported by most SMT solvers.

Stable Models and Linear Constraints

- A **level ranking** of a model M for a program P is a function $\mathrm{lr} : M \to \mathbb{N}$ such that for each $a \in M$, there is a rule r with $\mathrm{H}(r) = a$, $M \models \mathrm{B}(r)$ and for every $b \in \mathrm{B}^+(r)$, $\mathrm{lr}(a) - 1 \geq \mathrm{lr}(b)$ (or equivalently, $\mathrm{lr}(a) > \mathrm{lr}(b)$).
- **Example.** Consider a program P

$p_1 \leftarrow .$
$p_2 \leftarrow p_1 .$
$p_3 \leftarrow p_1 . \quad p_3 \leftarrow p_4 .$
$p_4 \leftarrow p_2 . \quad p_4 \leftarrow p_3 .$

Function $\mathrm{lr}_1(p_i) = i$ is a level ranking of $M = \{p_1, p_2, p_3, p_4\}$

Theorem (I.N, AMAI 2008)

Let M be a supported model of a finite normal program P. Then M is a stable model of P iff there is a level ranking of M for P.

Unique Rankings

- Stable models do not have unique level rankings.
- **Example.** For the program P

$p_1 \leftarrow .$	$M = \{p_1, p_2, p_3, p_4\}$
$p_2 \leftarrow p_1.$	has another level ranking
$p_3 \leftarrow p_1. \quad p_3 \leftarrow p_4.$	$\mathrm{lr}_2(p_1) = 1$,
$p_4 \leftarrow p_2. \quad p_4 \leftarrow p_3.$	$\mathrm{lr}_2(p_2) = \mathrm{lr}_2(p_3) = 2$,
	$\mathrm{lr}_2(p_4) = 3$.

- Level rankings can be made unique by adding two conditions:
  - unique lowest ranking level
  - no gaps

☞ **strong** level rankings. (I.N., AMAI2008)

Unique Rankings—cont'd

- A function $\mathrm{lr} : M \rightarrow \mathbb{N}$ is a **strong level ranking** of M for P iff for each $a \in M$ the following conditions hold:
  1. There is a rule $r \in P_M$ such that $\mathrm{H}(r) = a$ and for every $b \in \mathrm{B}^+(r)$, $\mathrm{lr}(a) - 1 \geq \mathrm{lr}(b)$.
  2. **If there is a rule $r \in P_M$ such that $\mathrm{H}(r) = a$ and $\mathrm{B}^+(r) = \emptyset$, then $\mathrm{lr}(a) = 1$.**
  3. **For every rule $r \in P_M$ such that $\mathrm{H}(r) = a$ there is $b \in \mathrm{B}^+(r)$ with $\mathrm{lr}(b) + 1 \geq \mathrm{lr}(a)$** (or equivalently $\mathrm{lr}(b) \geq \mathrm{lr}(a) - 1$).

  where $P_M = \{r \in P \mid M \models \mathrm{B}(r)\}$.
- For the program P and $M = \{p_1, p_2, p_3, p_4\}$,

$p_1 \leftarrow . \quad p_2 \leftarrow p_1.$	$\mathrm{lr}_1(p_i) = i$ is not a strong level
$p_3 \leftarrow p_1. \quad p_3 \leftarrow p_4.$	ranking because of $p_3 \leftarrow p_1.$
$p_4 \leftarrow p_2. \quad p_4 \leftarrow p_3.$	But $\mathrm{lr}_2(p_1) = 1$,
	$\mathrm{lr}_2(p_2) = \mathrm{lr}_2(p_3) = 2$,
	$\mathrm{lr}_2(p_4) = 3$ is.

Strong Rankings

(I.N., AMAI 2008):

- Every stable model has a strong level ranking.
- If there is a strong level ranking of M for P, then the ranking is a unique strong one.
- Strong level rankings are closely related to level numberings of rules and atoms used in (Janhunen 2004):
  - Every strong level ranking can be uniquely extended to rules to give a level numbering as defined in (Janhunen 2004).
  - Every level numbering as defined in (Janhunen 2004) when restricted to atoms is a strong level ranking.

Satisfiability Modulo Theories

- **Satisfiability Modulo Theories (SMT) problem**: a first-order theory T is given and the problem is to determine whether a formula **F is T-satisfiable** (whether $T \wedge F$ is satisfiable in the usual first-order sense).
- Some restrictions are typically assumed:
  - F is a **ground** (quantifier-free) formula that can contain **free constants** not in the signature of T but all other predicate and function symbols are in the signature of T.
  - **T-satisfiability of a conjunction** of such **ground literals** is **decidable**.

Example: EUF Logic

- Equality with Uninterpreted Functions
- The theory T consists of the axioms of reflexivity, symmetry, transitivity of '$=$' and for all function symbols f the monotonicity axiom $f(x_1,\ldots,x_n) = f(y_1,\ldots,y_n)$, if $x_i = y_i$ for all $i = 1,\ldots,n$
- The formula F could look like

$$\neg p \vee (b \neq c) \vee (f(b) \neq c) \vee (g(f(c)) \neq a) \vee (a = g(b))$$

  where p is a **new free predicate constant** (i.e. atomic proposition) and a, b, c are **free function constants** but f, g are function symbols in the signature of T.
- T-satisfiability of conjunctions of such ground literals is decidable by, e.g., congruence closure techniques. For example, $(b = c) \wedge (f(b) = c) \wedge g(f(c)) = a) \wedge (a \neq g(b))$ is not T-satisfiable.

Example: Difference Logic

- T is the theory of integers
- F is limited to contain only linear difference constraints of the form

$$x_i + k \geq x_j \text{ (or equivalently } x_j - x_i \leq k)$$

  where k is an arbitrary integer constant and $x_i, x_j \in \mathcal{X}$ are free constants (which can be seen as integer valued variables).
- **Difference logic = propositional logic + linear difference constraint**
- For example,

$$(x_1 + 2 \geq x_2) \leftrightarrow (p_1 \rightarrow \neg(x_2 - 3 \geq x_1))$$

  is a formula in difference logic where $2, 3$ are integer constants, x_1, x_2 **free function constants**, and p_1 **a free predicate constant**.

Difference Logic—cont'd

- A simplied semantics is given by a valuation τ consisting of a pair of functions $\tau_{\mathcal{P}} : \mathcal{P} \rightarrow \{\bot, \top\}$ and $\tau_{\mathcal{X}} : \mathcal{X} \rightarrow \mathbb{Z}$ where all other symbols (integer constants, $+, \geq$) are interpreted in the standard way.
- A valuation is extended to all formulas by applying the usual rules and by defining

$$\tau(x_i + k \geq x_j) = \top \text{ iff } \tau_{\mathcal{X}}(x_i) + k \geq \tau_{\mathcal{X}}(x_j)$$

- For example, given a valuation τ where $\tau_{\mathcal{X}}(x_1) = 1, \tau_{\mathcal{X}}(x_2) = 2, \tau_{\mathcal{P}}(p_1) = \bot$,
  - $\tau(x_1 + 2 \geq x_2) = \top$ as $(\tau_{\mathcal{X}}(x_1) + 2 =) 1 + 2 \geq 2 (= \tau_{\mathcal{X}}(x_2))$,
  - $\tau((x_1 + 2 \geq x_2) \leftrightarrow (p_1 \rightarrow \neg(x_2 - 3 \geq x_1))) = \top$

Difference Logic—cont'd

- Checking whether a set of linear constraints of the form $x_i + k \geq x_j$ is satisfiable can be decided in polynomial time. by reduction to finding a negative cycle in a weighted graph constructed from the constraints.
- Difference logic contains classical propositional logic as a special case.
- Deciding satisfiability in difference logic is NP-complete.
- Good theory propagation and explanation properties: ☞ efficient implementations in the DPLL(T) framework. (Nieuwenhuis, Oliveras & Tinelli, JACM 2006)

Translating LPs to Difference Logic

- The characterization of stable models using level rankings
  - Let M be a **supported model** of a finite normal program P. Then M is a stable model of P iff there is a level ranking of M for P.

  suggests a mapping $\mathrm{T}_{\mathrm{diff}}(P)$ of a logic program P to difference logic consisting of two parts:
  - completion $\mathrm{CC}(P)$ of P and
  - **ranking constraints** $\mathrm{R}(P)$.
- The completion $\mathrm{CC}(P)$:
  for an atom a having $k \geq 1$ rules in P, add the formula

$$a \leftrightarrow bd_a^1 \vee \cdots \vee bd_a^k$$

  and for each such rule a formula

$$bd_a^i \leftrightarrow b_1 \wedge \cdots \wedge b_m \wedge \neg c_1 \wedge \cdots \wedge \neg c_n$$

Ranking Constraints

- $\mathrm{R}(P)$: contains for each atom a which has $k \geq 1$ rules in P, a formula in difference logic

$$a \rightarrow \bigvee_{i=1}^{k} (bd_a^i \wedge (x_a - 1 \geq x_{b_1}) \wedge \cdots \wedge (x_a - 1 \geq x_{b_m}))$$

  where x_a, x_{b_i} are free function constants denoting the rankings of atoms a, b_i.

Example.

P:	$\mathrm{CC}(P)$:	$\mathrm{R}(P)$:
$p \leftarrow q, \mathrm{not}\ r.$	$\neg r$	$p \rightarrow (bd_p^1 \wedge (x_p - 1 \geq x_q))$
$q \leftarrow p, \mathrm{not}\ r.$	$p \leftrightarrow bd_p^1$	$q \rightarrow (bd_q^1 \wedge (x_q - 1 \geq x_p)).$
	$bd_p^1 \leftrightarrow q \wedge \neg r$	
	$q \leftrightarrow bd_q^1$	
	$bd_q^1 \leftrightarrow p \wedge \neg r$	

Valuations Capture Stable Models

Theorem (I.N., AMAI 2008)

- *If a set of atoms M is a stable model of a finite normal program P, then there is a satisfying valuation τ of $\mathrm{T}_{\mathrm{diff}}(P)$ such that $M = \{a \in \mathrm{At}(P) \mid \tau(a) = \top\}$.*
- *If there is a satisfying valuation τ of $\mathrm{T}_{\mathrm{diff}}(P)$, then $M = \{a \in \mathrm{At}(P) \mid \tau(a) = \top\}$ is a stable model of P.*

☞ A solver for difference logic can be used for computing stable models.

Example.

P:	$\mathrm{CC}(P)$:	$\mathrm{R}(P)$:
$p \leftarrow q, \mathrm{not}\ r.$	$\neg r$	$p \rightarrow (bd_p^1 \wedge (x_p - 1 \geq x_q))$
$q \leftarrow p, \mathrm{not}\ r.$	$p \leftrightarrow bd_p^1$	$q \rightarrow (bd_q^1 \wedge (x_q - 1 \geq x_p)).$
	$bd_p^1 \leftrightarrow q \wedge \neg r$	
	$q \leftrightarrow bd_q^1$	
	$bd_q^1 \leftrightarrow p \wedge \neg r$	

- $\mathrm{T}_{\mathrm{diff}}(P)$ has a satisfying valuation τ where $\tau(p) = \tau(q) = \bot$. Hence, P has a stable model $\{\}$.
- Note that there is no satisfying valuation τ where $\tau(p) = \tau(q) = \top$ because then also $\tau(x_p - 1 \geq x_q) = \tau(x_q - 1 \geq x_p) = \top$ should hold which is impossible.

Observations

- The translation is compact (of linear size).
- It uses a limited subset of difference logic:
  - Level rankings can be captured with constraints of the form $x_i - 1 \geq x_j$
- Strong level rankings can be translated to difference logic using additionally constraints of the form $x_i + 1 \geq x_j$ and $x_i \geq x_j$.
- The translation can be made even more compact and the number of required linear constraints can be reduced dramatically in typical cases by exploiting **strongly connected components** given by the **positive dependency graph** of the program (I.N., AMAI 2008).

Experiments

- A translator from ground programs to difference logic which supports a number of variants of the translation available (Janhunen & I.N. & Sevalnev, LPNMR 2009). `http://www.tcs.hut.fi/Software/lp2diff/`
- Any state-of-the-art SMT solver supporting difference logic can be used without modification as the backend solver.
- A number of variants also submitted to the ASP Competition 2009.
  ☞ The performance obtained by current (2009) SMT solvers (Z3, BARCELOGIC, YICES) surprisingly close to the best native ASP solvers (clasp).

Integrating ASP and SMT

- Here we demonstrate one straightforward approach to integration where ASP rules are extended with constraints supported by SMT solvers.
- First we consider the ground case with rules r of the form

$$a \leftarrow b_1, \ldots, b_m, \text{not } c_1, \ldots, \text{not } c_n, t_1, \ldots, t_l.$$

  where $\mathrm{B}^t(r) = \{t_1, \ldots, t_l\}$ is a set of ground theory literals (which can contain free constants).
- For defining the stable model semantics the theory atoms must be interpreted in a consistent way with the theory T.
- "Classical" interpretation for theory atoms.

Integrating ASP and SMT

- An interpretation of a program P is a pair (M, I) where M is a set of atoms and I is a set of ground theory atoms such that $T \wedge I \wedge \bar{I}$ is consistent where $\bar{I} = \{\neg t \mid t \text{ is theory atom in } P \text{ but } t \notin I\}$.
- Now an interpretation (M, I) of a program P is a **stable model** P iff
  (i) $(M, I) \models P$ and
  (ii) $M = LM(P^{(M,I)})$ where

$$P^{(M,I)} = \{\mathrm{H}(r) \leftarrow \mathrm{B}^+(r) \mid r \in P, \mathrm{B}^-(r) \cap M = \emptyset, I \models \mathrm{B}^t(r)\}.$$

Example

- Consider the case where we use the theory of integers and allow linear constraints as theory atoms.
- Program P:
  $\leftarrow$ not s.
  $s \leftarrow x > z$.
  $p \leftarrow x \leq y$.
  $p \leftarrow q$.
  $q \leftarrow p, y \leq z$.

  $\mathcal{M}_1 = (\{s\}, \{x > z\})$ is a stable model of P:
  - $(x > z) \wedge \neg(x \leq y) \wedge \neg(y \leq z)$ is T-consistent
  - $\mathcal{M}_1 \models P$ and
  - $\{s\} = LM(P^{\mathcal{M}_1})$ where $P^{\mathcal{M}_1} = \{s \leftarrow .\ \ p \leftarrow q.\}$.
- $\mathcal{M}_2 = (\{s, p, q\}, \{x > z, x \leq y, y \leq z\})$ is not a stable model because $(x > z) \wedge (x \leq y) \wedge (y \leq z)$ is not T-consistent.
- $\mathcal{M}_3 = (\{s, p, q\}, \{x > z, y \leq z\})$ is not a stable model as $\{s, p, q\} \neq LM(P^{\mathcal{M}_3})$ with $P^{\mathcal{M}_3} = \{s \leftarrow .\ p \leftarrow q.\ q \leftarrow p.\}$

Embedding to SMT

- Assume that we are given an SMT solver supporting a logic containing difference logic.
- Consider now a class of rules where ground theory literals supported by the solver are allowed.
- For this class of rules it is straightforward to develop a translation to the logic supported by the solver.
- In fact we can use the translation described above with the following extension in the completion:
- For a rule r of the form

$$a \leftarrow b_1, \ldots, b_m, \text{not } c_1, \ldots, \text{not } c_n, t_1, \ldots, t_l.$$

the formula capturing the satisfaction of the body is now

$$bd_a^i \leftrightarrow b_1 \wedge \cdots \wedge b_m \wedge \neg c_1 \wedge \cdots \wedge \neg c_n \wedge t_1 \wedge \cdots \wedge t_l$$

The non-ground case

- The non-ground case is handled in the usual way by treating a rule with variables as a shorthand for the set of its Herbrand instantiations.
- To support the interaction between the regular and theory literals, an indexing technique can be introduced: **free constants** in the ground theory atoms can be **indexed by Hebrand terms**
- For example,

$$\leftarrow occurs(a, S_1), occurs(b, S_2), t(S_2) - t(S_1) > 7$$

is a shorthand for a set of ground rules

$$\leftarrow occurs(a, s_1), occurs(b, s_2), t(s_2) - t(s_1) > 7$$

where s_1, s_2 range over Herbrand terms and $t(s_1), t(s_2)$ are treated as **free constants** of the background theory.

The non-ground case

- For guaranteeing finite grounding a domain (or range) restriction can be used for the (index) variables, too: each variable in a rule occurs in some positive regular body atom.
- With the indexing technique mixed atoms and related semantical complications are avoided.
- For example, it is easy to express typical scheduling constraints
  $\leftarrow next(S_1, S_0), t(S_1) < t(S_0)$
  $\leftarrow goal(S), t(S) - t(0) > 60$
  $\leftarrow next(S_1, S_0), occur(goto(john, home), S_0),$
  $holds(atloc(john, office), S_0), t(S_1) - t(S_0) < 20$
  used when mixing planning and scheduling (Mellarkod, Gelfond & Zhang, AMAI 2008).

Conclusions

- Difference logic allows for a compact translation of rules.
- The translation to difference logic opens up the possibility of using difference logic solvers as a computational platform for implementing ASP.
- The performance obtained by the translation and current SMT solvers is already surprisingly close to the best state-of-the-art ASP solvers.
- The translation makes it possible to embed rule-based reasoning directly into SMT systems that support difference logic.
- An interesting approach to integrating ASP and SMT.

References

- I.N. Stable Models and Difference Logic. Annals of Mathematics and Artificial Intelligence 1-4 (2008) 53, 313-329.
- T. Janhunen, I.N., M. Sevalnev. Computing Stable Models via Reductions to Difference Logic. LPNMR 2009.