Aalto University
School of Science

Tight Integration of Non-Ground Answer Set Programming and Satisfiability Modulo Theories

Tomi Janhunen, Guohua Liu, and Ilkka Niemelä

Aalto University, Department of Information and Computer Science

GTTV'11, May 16, 2011

Background

- *Non-Boolean variables* are important primitives in logical modeling in a number of disciplines: ASP, CP, . . .
- The SMT framework enriches Boolean satisfiability checking in terms of a *background theory*.
- Logic programs under answer sets can be translated into an SMT fragment, viz. *difference logic* (DL).
  - Niemelä [AMAI, 2008]
  - Janhunen, Niemelä, and Sevalnev [LPNMR, 2009]
- Translations in the other direction are impeded by the potentially infinite domains of variables involved.
- There are existing approaches that combine ASP and CP:
  - Balduccini [ASPOCP, 2009]
  - Gebser et al. [ICLP, 2009]
  - Mellarkord et al. [AMAI, 2008]

Objectives

- Our goal is to *integrate* ASP and SMT so that non-Boolean variables of these formalisms can be used together.
- We aim at a rule-based language ASP(SMT) which is enriched by *theory atoms* from a particular SMT dialect.

Example

Let us formalize the n-queens problem in ASP(DL):

$$\begin{aligned}
&\text{queen}(1..n).\\
&\text{int}(row(X)) \leftarrow \text{queen}(X).\\
&\leftarrow row(X) - row(Y) = 0,\ \text{queen}(X),\ \text{queen}(Y),\ X < Y.\\
&\ldots
\end{aligned}$$

Here $row(X) - row(Y) = 0$ is a theory atom from DL.

Outline

Preliminaries

Integrated Language

Problem Modeling

Implementation

Preliminary Experiments

Conclusion

PRELIMINARIES

- A *normal logic program* P is a set of rules of the form

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

parts of which are abbreviated in the sequel by

$$\begin{aligned} hd(r) &= a, \\ bd^+(r) &= \{b_1, \ldots, b_m\}, \text{ and} \\ bd^-(r) &= \{c_1, \ldots, c_n\}. \end{aligned}$$

- An interpretation $M \subseteq \mathrm{At}(P)$ is an *answer set* of P iff M is the least model of the Gelfond-Lifschitz [1988] *reduct*

$$P^M = \{hd(r) \leftarrow bd^+(r) \mid r \in P \text{ and } bd^-(r) \cap M = \emptyset\}.$$

Weight Constraint Programs

- The extended rule types of the SMODELS system are based on *cardinality* and *weight constraints* of the form

$$l\{b_1,\ldots,b_n,\text{not } c_1,\ldots,\text{not } c_m\}u$$
$$l\{b_1 = w_{b_1},\ldots,b_n = w_{b_n},\text{not } c_1 = w_{c_1},\ldots,\text{not } c_m = w_{c_m}\}u$$

- The class of weight constraint programs is based on rule elements of this kind instead of propositional atoms.
- Extended rule types can be translated back to normal programs using grounders and a tool called LP2NORMAL[1].
- It is also straightforward to translate ground weight constraints into DL as part of the ASP to DL translation.

[1] `http://www.tcs.hut.fi/Software/asptools/`

Difference Logic

- *Difference logic* (DL) extends classical propositional logic with atomic *difference constraints* of the form

$$x - y \leq k.$$

- Operators $<$, $>$, $\geq$, $=$, and $\neq$ can also be used.
- A difference constraint $x - y \leq k$ is *satisfied* in an *interpretation* $\langle I, \tau \rangle$, denoted $\langle I, \tau \rangle \models x - y \leq k$, iff

$$\tau(x) - \tau(y) \leq k.$$

- Other primitives are covered by propositional logic.

Example

The theory $\{x - y < 0,\ y - z < 0,\ z - x < 0\}$ is unsatisfiable.

INTEGRATED LANGUAGE

- A program P in ASP(DL) consists of rules 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 $t_1, \ldots, t_l$ are difference constraints.
- An *interpretation* of P is defined as a pair $\langle I, T \rangle$ such that $T \cup \{\neg t \mid t \text{ appears in } P,\ t \notin T\}$ is satisfiable in DL.
- A model M of a program P is an *answer set* of P iff the *propositional part* M^p is the least model of

$$P^M = \{\ hd(r) \leftarrow bd^+(r) \mid r \in P,\ bd^-(r) \cap M^p = \emptyset,\ \text{and } M^t \models bd^t(r)\ \}.$$

Example

Consider the following ASP(DL) program P:

$$\leftarrow \text{not } s. \quad s \leftarrow x > z. \quad p \leftarrow x \leq y. \quad p \leftarrow q. \quad q \leftarrow p, y \leq z.$$

1. $M_1 = (\{s\}, \{x > z\})$ is an answer set of P since $\{(x > z), \neg(x \leq y), \neg(y \leq z)\}$ is satisfiable in DL, $M_1 \models P$, and $\{s\}$ is the least model of $P^{M_1} = \{s \leftarrow;\ p \leftarrow q\}$.
2. $M_2 = (\{s, p, q\}, \{x > z, x \leq y, y \leq z\})$ is not an answer set, since $\{(x > z), (x \leq y), (y \leq z)\}$ is not satisfiable.
3. $M_3 = (\{s, p, q\}, \{x > z, y \leq z\})$ is not an answer set as the least model of $P^{M_3} = \{s \leftarrow;\ p \leftarrow q;\ q \leftarrow p\}$ is $\{s\}$.

PROBLEM MODELING

- We provide a number of examples to illustrate the use and advantages of ASP(DL) in logical modelling.
- In certain cases, difference constraints enable much more *concise* encoding than a pure ASP language.
- In the paper, we provide sample encodings for
  1. a scheduling problem,
  2. routing under time constraints,
  3. trip planning, and
  4. a sorting problem.
- Savings up to quadratic factors (e.g., $|T|^2$) are perceived.

Scheduling Problem

- Predicate read(P, N, D) represents the duration D of a person P reading a newspaper N.
- Integer variables $s(P, N)$ and $e(P, N)$ capture the respective starting and ending times.

$$\leftarrow \quad e(P,N) - s(P,N) \neq D, \quad \text{read}(P,N,D).$$

$$\leftarrow \quad s(P,N_1) < s(P,N_2), \quad s(P,N_2) - s(P,N_1) < D_1,$$
$$\text{read}(P,N_1,D_1), \quad \text{read}(P,N_2,D_2), \quad N_1 \neq N_2.$$

$$\leftarrow \quad s(P_1,N) < s(P_2,N), \quad s(P_2,N) - s(P_1,N) < D_1,$$
$$\text{read}(P_1,N,D_1), \quad \text{read}(P_2,N,D_2), P_1 \neq P_2.$$

$$\leftarrow \quad e(P,N) > \mathit{deadline}, \quad \text{read}(P,N,D).$$

Sorting Problem

- The goal is to sort given numbers in an increasing order:

$$\leftarrow \quad p(X_1) = p(X_2),\ \ X_1 \neq X_2,\ \ \text{number}(X_1; X_2).$$
$$\leftarrow \quad X_1 > X_2,\ \ p(X_1) < p(X_2),\ \ \text{number}(X_1; X_2).$$

- The respective ASP encoding is as follows:

$$1\{\text{place}(X, Y) : \text{position}(Y)\}1 \leftarrow \text{number}(X).$$
$$1\{\text{place}(X, Y) : \text{number}(X)\}1 \leftarrow \text{position}(Y).$$
$$\leftarrow \quad X_1 > X_2, Y_1 < Y_2,\ \ \text{place}(X_1, Y_1; X_2, Y_2),$$
$$\text{number}(X_1; X_2),\ \ \text{position}(Y_1, Y_2).$$

IMPLEMENTATION

- Theory atoms are represented with special predicates like

$$\text{dl\_lt}(X, Y, D)$$

  for the difference constraint $x - y < d$ in DL.
- Special domain predicates such as $\text{int}(V)$ for DL are used to declare the domains of theory constants.
- Names of these predicates can be easily changed.

Example

$$\text{int}(at(X)) \leftarrow \text{edge}(X, Y, W).$$
$$\text{int}(at(Y)) \leftarrow \text{edge}(X, Y, W).$$
$$\leftarrow \text{route}(X, Y), \text{edge}(X, Y, W), \text{dl\_lt}(at(Y), at(X), W).$$

Main Steps

A shell script called DINGO[2]

1. grounds a logic program using GRINGO which treats theory atoms as *externally defined* predicates,
2. extracts the relevant type information from the ground program to create the prologue of the DL theory,
3. translates the ground program into DL using LP2DIFF,
4. extracts the ground instances of theory atoms and expands them into DL using a standard macro processor M4, and
5. invokes an SMT solver to compute a satisfying assignment (if any) and extracts an answer set.

---

[2] `http://www.tcs.hut.fi/Software/lp2diff/`

Live Demo

```
$ cat queen.lp
```

Live Demo

```
$ cat queen.lp

queen(1..n).
int(row(X)) :- queen(X).
int(zero).

:- dl_le(row(X),zero,0), queen(X).
:- dl_gt(row(X),zero,n), queen(X).
:- dl_eq(row(X),row(Y),0), queen(X;Y), X<Y.
:- dl_eq(row(X),row(Y),#abs(X-Y)), queen(X;Y), X!=Y.
```

Live Demo

```
$ cat queen.lp

queen(1..n).
int(row(X)) :- queen(X).
int(zero).

:- dl_le(row(X),zero,0), queen(X).
:- dl_gt(row(X),zero,n), queen(X).
:- dl_eq(row(X),row(Y),0), queen(X;Y), X<Y.
:- dl_eq(row(X),row(Y),#abs(X-Y)), queen(X;Y), X!=Y.

$ dingo.sh -cn=7 queen.lp
```

Live Demo

```
$ cat queen.lp

queen(1..n).
int(row(X)) :- queen(X).
int(zero).

:- dl_le(row(X),zero,0), queen(X).
:- dl_gt(row(X),zero,n), queen(X).
:- dl_eq(row(X),row(Y),0), queen(X;Y), X<Y.
:- dl_eq(row(X),row(Y),#abs(X-Y)), queen(X;Y), X!=Y.

$ dingo.sh -cn=7 queen.lp

SATISFIABLE
queen(1) queen(2) queen(3) queen(4) queen(5) queen(6) queen(7)
Theory:
Vars: row(7)=7 zero=0 row(6)=2 row(5)=4 row(4)=6 row(3)=1 row(2)
```

PRELIMINARY EXPERIMENTS

- We compared two systems, DINGO and CLINGO (v. 2.0.3), using encodings in ASP(DL) and pure ASP, respectively.
- Running times of these system and the sizes of resulting ground instances were subject to comparison.
- For each problem, we experimented with 10 groups of instances and each group contained 100 randomly generated instances.
- The cut off time was set to 300 seconds.

Newspaper Benchmark

Deadline	DINGO		CLINGO	
	time	size ratio	time	size ratio
100	0.09	1.0	2.10	1.0
200	0.11	1.1	9.00	3.1
300	0.11	1.3	21.32	6.3
400	0.10	1.4	36.68	15
500	0.12	1.5	61.15	23
600	0.12	1.7	93.51	34
700	0.11	1.8	–	44
800	0.11	1.9	–	60
900	0.12	2.1	–	74
1000	0.13	2.2	–	81

Sorting Benchmark

Numbers	DINGO		CLINGO	
	time	size ratio	time	size ratio
60	0.59	1.0	13.12	1.0
70	0.77	1.3	25.50	2.1
80	1.01	1.8	49.70	2.7
90	1.26	2.3	76.14	4.9
100	1.54	2.8	145.43	7.8
110	1.84	3.4	–	12
120	2.25	4.1	–	17
130	2.71	4.8	–	28
140	3.17	5.6	–	34
150	3.56	6.4	–	38

CONCLUSION

- In this research, we present an approach to integrating the languages used in ASP and SMT.
- The idea is to enrich rules with extra conditions which together form a theory in a particular SMT fragment.
- Our prototype implementation, viz. DINGO, exploits off-the-shelf ASP and SMT components for grounding (GRINGO) and the search for answer sets (Z3).
- Macros enable the use of standard ASP grounders for the creation of SMT theories of interest.
- The combined language ASP(DL) enables more concise ways to encode problems from various domains.
- Our first experiments using these encodings also suggest positive effects on solving time.

Future Work

- Other SMT dialects should be taken into consideration.
- It may be necessary to develop a translation from ASP into the SMT dialect in question and to extend LP2DIFF.
- The optimization of answer-set programs involving external/input atoms requires special attention.
- There is also potential for combining SMT dialects as long as there is a suitable target language and a back-end SMT solver available.
- It is also feasible to develop ASP(SMT) encodings in a modular way and to link modules together with LPCAT.