Aalto University
School of Science

Answer Set Programming as SAT modulo Acyclicity

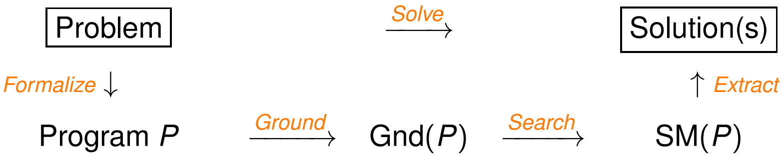
Martin Gebser, Tomi Janhunen, and Jussi Rintanen

Helsinki Institute for Information Technology HIIT
Department of Information and Computer Science
Aalto University
Finland

ECAI 2014, Prague, Czech Republic, August 20

Answer Set Programming

Answer set programming (ASP) features a rule-based syntax subject to answer-set semantics.



Some native answer set solvers:

- CLASP `http://potassco.sourceforge.net/`
- CMODELS `http://www.cs.utexas.edu/~tag/cmodels/`
- DLV `http://www.dlvsystem.com/`
- IDP^3 `http://dtai.cs.kuleuven.be/krr/software/idp/`
- SMODELS `http://research.ics.aalto.fi/software/`

Example: SuDoku Puzzle

1	9	3	8	6	7	4	2	5
4	6	8	5	3	2	9	1	7
7	5	2	1	4	9	6	8	3
6	2	1	4	7	3	5	9	8
5	3	4	9	1	8	7	6	2
9	8	7	2	5	6	3	4	1
2	1	6	3	9	5	8	7	4
8	7	5	6	2	4	1	3	9
3	4	9	7	8	1	2	5	6

```
number(1..9).
border(1). border(4). border(7).
region(X,Y) :- border(X), border(Y).

1 { value(X,Y,N):number(X):number(Y):
    X1<=X: X<=X1+2: Y1<=Y: Y<=Y1+2 } 1
  :- number(N), region(X1,Y1).

:- 2 {value(X,Y,N):number(N)}, number(X), number(Y).
:- 2 {value(X,Y,N):number(Y)}, number(N), number(X).
:- 2 {value(X,Y,N):number(X)}, number(N), number(Y).
```

Example: Running the Solver

```
$ gringo sudoku.lp royle.lp | clasp 0
clasp version 3.0.4
Reading from stdin
Solving...
Answer: 1
value(1,3,2) value(1,9,1) value(2,2,7) value(2,5,3) value(3,5,4)
value(3,7,2) value(4,4,2) value(5,7,4) value(5,8,3) value(6,1,1)
value(6,3,5) value(6,4,6) value(7,5,7) value(8,2,3) value(9,4,1)
value(9,9,5) value(1,2,9) value(3,1,8) ...
Answer: 2
value(1,3,2) value(1,9,1) value(2,2,7) value(2,5,3) value(3,5,4)
value(3,7,2) value(4,4,2) value(5,7,4) value(5,8,3) value(6,1,1)
value(6,3,5) value(6,4,6) value(7,5,7) value(8,2,3) value(9,4,1)
value(9,9,5) value(3,1,9) ...
SATISFIABLE
Models       : 2
...
```

Key Features of ASP

- Typical ASP encodings follow a three-phase design:
  1. Generate the solution candidates
  2. Define the required concepts
  3. Test if a candidate satisfies its criteria
- Default negation favors concise encodings.
- Basic database operations are definable in terms of rules:
  - Projection: $\text{node}(X) \leftarrow \text{edge}(Y, X).$
  - Union: $\text{node}(X) \leftarrow \text{edge}(Y, X).$ $\quad \text{node}(Y) \leftarrow \text{edge}(Y, X).$
  - Intersection: $\text{symm}(X, Y) \leftarrow \text{edge}(X, Y), \text{edge}(Y, X).$
  - Complement: $\text{unidir}(X, Y) \leftarrow \text{edge}(X, Y), \text{not edge}(Y, X).$
- Recursive definitions are also supported:

$$\text{path}(X, Y) \leftarrow \text{edge}(X, Z), \text{path}(Z, Y), \text{node}(Y).$$

Translation-Based ASP

ASP can be implemented by translating ground programs into:

- Boolean Satisfiability (SAT)
  [J., ECAI, 2004; J. and Niemelä, MG-65, 2010]
- Integer Difference Logic (IDL)
  [Niemelä, AMAI, 2008; J. et al., LPNMR, 2009]
- Integer Programming (IP)
  [Liu et al., KR, 2012]
- Bit-Vector Logic (BV)
  [Nguyen et al., INAP, 2011; Extended in 2013]

☞ Existing solver technology can be harnessed for ASP!

Motivation

- Complexities of translations vary in program length n:

$\mathcal{O}(n)$	IDL, IP, BV	
$\mathcal{O}(n \times \log_2 n)$	SAT	[J., ECAI 2004]
$\mathcal{O}(n^2)$	SAT	[Lin & Zhao, IJCAI 2003]
$\mathcal{O}(2^n)$	SAT	[Lin & Zhao, AIJ 2004]

- What would be a minimal extension of SAT such that
  1. a linear embedding from ASP is enabled and
  2. the extension is efficiently implementable?
- In this paper, we consider embeddings into an extension based on graphs subject to an acyclicity constraint:

  M. Gebser, T. Janhunen, and J. Rintanen:
  "*Satisfiability Modulo Graphs: Acyclicity*" [JELIA 2014].

Outline

Formalisms of Interest

Translating Programs into SAT modulo Acyclicity

Implementation and Experiments

Conclusion

Source Formalism: Normal Programs

- Normal logic programs (NLPs) consist of rules of the form:

$$a \leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m.$$

- The semantics is given by stable models, also known as answer sets, satisfying [Gelfond and Lifschitz, ICLP, 1988]:

$$M = \text{LM}(P^M).$$

Example

Consider the following program:

$$a \leftarrow b. \quad a \leftarrow c. \quad b \leftarrow a. \quad c \leftarrow \text{not } d. \quad d \leftarrow \text{not } c.$$

$\Longrightarrow$ $M_1 = \{a, b, c\}$ is stable but $M_2 = \{a, b, d\}$ is not.

Target Formalism: Syntax

A theory in SAT modulo acyclicity (ACYC) is a tuple $\langle X, C, N, A, l \rangle$ where

1. C is a set of clauses based on propositional variables in X,
2. $G = \langle N, A \rangle$ is a directed graph with a finite set of nodes N and arcs $A \subseteq N \times N$, and
3. $l : A \to X$ is a labeling that assigns a propositional variable $l(u, v)$ to every arc $\langle u, v \rangle \in A$ in the graph G.

Example

Rewriting our NLP using $N = \{a, b\}$ and $E = \{\langle a, b \rangle, \langle b, a \rangle\}$:

$$a \lor \neg b, \quad a \lor \neg c, \quad \neg a \lor b \lor c, \quad b \lor \neg a, \quad \neg b \lor a,$$
$$c \lor d, \quad \neg c \lor \neg d, \quad \neg a \lor c \lor e_{\langle a,b \rangle}, \quad \neg b \lor e_{\langle b,a \rangle}.$$

Target Formalism: Semantics

An ACYC theory $T = \langle X, C, N, A, l \rangle$ is satisfied by an interpretation $M \subseteq X$, denoted $M \models T$, iff

1. $M \models C$ and
2. $\langle N, A_M \rangle$ with $A_M = \{\langle u, v \rangle \in A \mid M \models l(u, v)\}$ is *acyclic*.

Example

Recall the theory T from our running example:

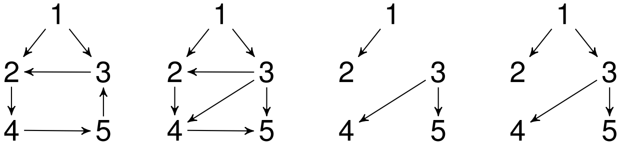
$$a \lor \neg b, \quad a \lor \neg c, \quad \neg a \lor b \lor c, \quad b \lor \neg a, \quad \neg b \lor a,$$
$$c \lor d, \quad \neg c \lor \neg d, \quad \neg a \lor c \lor e_{\langle a,b \rangle}, \quad \neg b \lor e_{\langle b,a \rangle}.$$

$$\Longrightarrow \quad M_1 = \{a, b, c, e_{\langle b,a \rangle}\} \models T \quad \text{but}$$
$$M_2 = \{a, b, d, e_{\langle a,b \rangle}, e_{\langle b,a \rangle}\} \not\models T.$$

Applications in Sight

Acyclicity constraints lend themselves for many purposes:

- Specifying a variety of topological structures:
  - Trees and forests (both directed and undirected)
  - Directed acyclic graphs (DAGs)
  - Chordal graphs



- Hamiltonian cycles
- Formalizing paths and reachability in general

General Translation from ASP to ACYC

- The classical models of the completion Comp(P) coincide with the supported models P [Apt et al., 1988].
- The strong groundedness of stable models can be captured by assigning numbers/ordinals to atoms [Elkan, AIJ 1990; Fages, JMLCS 1994; Erdem & Lifschitz, TPLP 2003].
- We follow the linear translation into IDL based on level rankings [Niemelä, AMAI 2008; J. et al., LPNMR 2009].
- The translation has to be applied only to atoms $a \in \mathrm{At}(P)$ having a non-trivial component $\mathrm{SCC}(a)$ with $|\mathrm{SCC}(a)| > 1$.

In our running example, we have $\mathrm{SCC}(a) = \{a, b\} = \mathrm{SCC}(b)$:

$$a \leftarrow b. \quad a \leftarrow c. \quad b \leftarrow a. \quad c \leftarrow \mathrm{not}\ d. \quad d \leftarrow \mathrm{not}\ c.$$

Identifying Rule Bodies

Following [Tseitin, 1968], the body $\mathrm{B}(r)$ of a defining rule $r \in \mathrm{Def}_P(a)$ is given a new name bd_r by

1. the clause $\mathrm{bd}_r \lor \bigvee_{b \in \mathrm{B}^+(r)} \neg b \lor \bigvee_{c \in \mathrm{B}^-(r)} c$,
2. for each $b \in \mathrm{B}^+(r)$, the clause $\neg \mathrm{bd}_r \lor b$, and
3. for each $c \in \mathrm{B}^-(r)$, the clause $\neg \mathrm{bd}_r \lor \neg c$.

$\Longrightarrow$ Effectively, we have $\mathrm{bd}_r \leftrightarrow \bigwedge_{b \in \mathrm{B}^+(r)} b \land \bigwedge_{c \in \mathrm{B}^-(r)} \neg c$.

Example

Rule:	$a \leftarrow b.$	$a \leftarrow c.$	$b \leftarrow a.$
Translation:	$\mathrm{bd}_1 \lor \neg b$	$\mathrm{bd}_2 \lor \neg c$	$\mathrm{bd}_3 \lor \neg a$
	$\neg \mathrm{bd}_1 \lor b$	$\neg \mathrm{bd}_2 \lor c$	$\neg \mathrm{bd}_3 \lor a$

Well Support from Internal Rules

For the well-support provided by a rule $r \in \mathrm{IDef}_P(a)$:

1. The clause $\mathrm{ws}_r \vee \neg\mathrm{bd}_r \vee \bigvee_{b \in \mathrm{B}^+(r) \cap \mathrm{SCC}(a)} \neg\mathrm{e}_{\langle a,b \rangle}$.
2. The clause $\neg\mathrm{ws}_r \vee \mathrm{bd}_r$.
3. For each $b \in \mathrm{B}^+(r) \cap \mathrm{SCC}(a)$, the clause $\neg\mathrm{ws}_r \vee \mathrm{e}_{\langle a,b \rangle}$.

$\Longrightarrow$ Effectively, we have $\mathrm{ws}_r \leftrightarrow \mathrm{bd}_r \wedge \bigwedge_{b \in \mathrm{B}^+(r) \cap \mathrm{SCC}(a)} \mathrm{e}_{\langle a,b \rangle}$.

Example

Internal rule:	$a \leftarrow b.$	$b \leftarrow a.$
Translation:	$\mathrm{ws}_1 \vee \neg\mathrm{bd}_1 \vee \neg\mathrm{e}_{\langle a,b \rangle}$ $\neg\mathrm{ws}_1 \vee \mathrm{bd}_1$ $\neg\mathrm{ws}_1 \vee \mathrm{e}_{\langle a,b \rangle}$	$\mathrm{ws}_3 \vee \neg\mathrm{bd}_3 \vee \neg\mathrm{e}_{\langle b,a \rangle}$ $\neg\mathrm{ws}_3 \vee \mathrm{bd}_3$ $\neg\mathrm{ws}_3 \vee \mathrm{e}_{\langle b,a \rangle}$

Enforcing Support for Atoms

For the definition $\mathrm{Def}_P(a)$ of an atom a in a program P:

1. For each $r \in \mathrm{Def}_P(a)$, the clause $a \vee \neg \mathrm{bd}_r$.
2. The clause $\neg a \vee \bigvee_{r \in \mathrm{EDef}_P(a)} \mathrm{bd}_r \vee \bigvee_{r \in \mathrm{IDef}_P(a)} \mathrm{ws}_r$.

$\Longrightarrow$ Effectively, this entails that $a \leftrightarrow \bigvee_{r \in \mathrm{Def}_P(a)} \mathrm{B}(r)$.

Example

Definition:	$a \leftarrow b.$ $\quad a \leftarrow c.$	$b \leftarrow a.$
Translation:	$a \vee \neg \mathrm{bd}_1,$ $\quad a \vee \neg \mathrm{bd}_2$ $\neg a \vee \mathrm{ws}_1 \vee \mathrm{bd}_2$	$b \vee \neg \mathrm{bd}_3$ $\neg b \vee \mathrm{ws}_3$

Overall Properties of the Translation

- The resulting translation $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ of a normal program P is linear in the length of P.
- A one-to-many correspondence between the stable models of P and the models of $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ is obtained.

Proposition

Let P be a normal logic program and $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ its translation into SAT modulo acyclicity.

1. *If $M \in \mathrm{SM}(P)$, then there is a model $N \models \mathrm{Tr}_{\mathrm{ACYC}}(P)$ such that $M = N \cap \mathrm{At}(P)$.*
2. *If $N \models \mathrm{Tr}_{\mathrm{ACYC}}(P)$, then $M \in \mathrm{SM}(P)$ for $M = N \cap \mathrm{At}(P)$.*

Extension: Disabling Edges Dynamically

- An edge variable $e_{\langle a,b \rangle}$ can be falsified if
  1. a is known to be false,
  2. a has an externally supporting rule, or
  3. a has an internally supporting rule $r \in \mathrm{IDef}_P(a)$ such that $b \notin \mathrm{B}^+(r)$.
- The extended translation $\mathrm{Tr}^+_{\mathrm{ACYC}}(P)$ gives rise to a similar but tighter correspondence of models.

Example

Definition:	$a \leftarrow b. \quad a \leftarrow c.$	$b \leftarrow a.$
Case 1:	$a \vee \neg e_{\langle a,b \rangle}$	$b \vee \neg e_{\langle b,a \rangle}$
Case 2:	$\neg bd_2 \vee \neg e_{\langle a,b \rangle}$	–
Case 3:	–	–

Implementation

- For tool interoperability, the SMODELS format is used as an intermediate format for representing ground programs.
- Extended rules, such as choice, cardinality, and weight rules may have to be translated away using LP2NORMAL2.
- To enable cross-translation for different back-end solvers,
  1. the input program is instrumented with auxiliary atoms and auxiliary rules corresponding to $\mathrm{Tr}^{+}_{\mathrm{ACYC}}$ and
  2. the completion is produced in the target format of interest.
- Our tools produce a number of output formats:
  1. DIMACS with optional ACYC and MAXSAT extensions
  2. SMT Library 2.0 (QF_IDL and QF_BV fragments)
  3. PB format
  4. CPLEX

Tool Support

<table>
<tr><td colspan="2">gringo / lparse</td></tr>
<tr><td colspan="2">lpstrip</td></tr>
<tr><td colspan="2">lpcat</td></tr>
<tr><td>lp2normal2</td><td>–</td></tr>
<tr><td colspan="2">lp2acyc</td></tr>
<tr><td>lp2sat
[-g]</td><td>acyc2solver
[--diff]
[--bv]
[--pb]
[--mip]</td></tr>
</table>

The tool collection is published under:

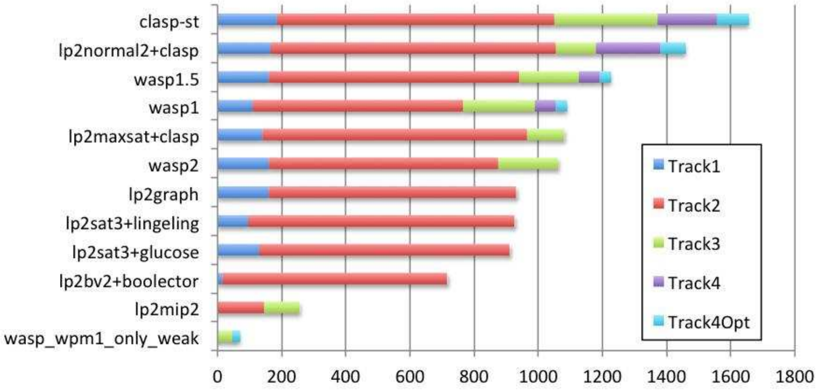
`http://research.ics.aalto.fi/software/asp/lp2acyc/`

Experiments

Problem	Hamilton		Tree			
Size	100	150	25	50	75	100
CLASP	0.95	20.16	4.37	1193.09	1495.32	1995.19
ACYCGLUCOSE	0.07	0.15	**0.74**	315.83	999.07	1414.68
ACYCMINISAT	**0.04**	**0.12**	0.83	544.43	1025.02	1224.28
Z3	2.45	50.64	4.75	1208.36	1726.56	2538.20
ACYCGLUCOSE-Tr_{ACYC}	0.93	13.75	1.40	271.93	973.22	1388.82
ACYCMINISAT-Tr_{ACYC}	0.76	7.28	0.80	484.92	879.18	1030.79
Z3-Tr_{ACYC}	35.80	331.11	6.30	1178.44	2266.66	2714.01
ACYCGLUCOSE-Tr^{+}_{ACYC}	**0.04**	0.18	1.09	**264.28**	931.28	1379.15
ACYCMINISAT-Tr^{+}_{ACYC}	0.08	0.32	0.77	473.64	**852.78**	**1016.50**
Z3-Tr^{+}_{ACYC}	27.72	239.83	7.03	1230.51	1976.20	2562.70

ASP Competition 2014

The LP2GRAPH system was based on the translation $\mathrm{Tr}^{+}_{\mathrm{ACYC}}$ and using ACYCGLUCOSE as the back-end solver.



[https://www.mat.unical.it/aspcomp2014/]

Conclusion

- Translation-based ASP aims to exploit
  - the expressive power of ASP and
  - the potential behind existing solver technology.
- The translation from ASP into SAT modulo acyclicity
  - is linear and
  - preserves stable models up to original signature.
- The cross-translation of ASP is enabled by
  - a suitable intermediate format and
  - postponing format-specific aspects to the last step.
- Future extensions:
  - Support for further formats and solver types
  - Covering optimization more widely