Aalto University
School of Science

Answer Set Programming modulo Acyclicity

Jori Bomanson[1], Martin Gebser[1,2], Tomi Janhunen[1]
Benjamin Kaufmann[2], and Torsten Schaub[2,3]

[1] Aalto University, Finland
[2] University of Potsdam, Germany
[3] INRIA Rennes, France

LPNMR'15, Lexington, Kentucky, USA, September 30, 2015

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]

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]

— SAT modulo Acyclicity (ACYC-SAT)
  [G. et al., ECAI 2014]

Extensions to ASP

- There are existing SMT-style extensions of ASP:
  - Constraint programming [G. et al., ICLP 2009]
  - Difference logic [J. et al., GTTV 2011]
  - Linear programming [Liu et al., INAP 2013]
  - General SMT [Lee & Meng, IJCAI 2013]

Extensions to ASP

- There are existing SMT-style extensions of ASP:
  - Constraint programming [G. et al., ICLP 2009]
  - Difference logic [J. et al., GTTV 2011]
  - Linear programming [Liu et al., INAP 2013]
  - General SMT [Lee & Meng, IJCAI 2013]

- In this work, we propose ASP modulo Acyclicity
  - as an extension to ASP and
  - as a target formalism for translations of ASP.

Extensions to ASP

- There are existing SMT-style extensions of ASP:
  - Constraint programming [G. et al., ICLP 2009]
  - Difference logic [J. et al., GTTV 2011]
  - Linear programming [Liu et al., INAP 2013]
  - General SMT [Lee & Meng, IJCAI 2013]

- In this work, we propose ASP modulo Acyclicity
  - as an extension to ASP and
  - as a target formalism for translations of ASP.

- Functionality available in CLASP version 3.2.0 onward.

Standard Logic Programs

- Logic programs consist of rules of the following forms:

$$
\begin{aligned}
a &\leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m. \\
\{a\} &\leftarrow b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m. \\
a &\leftarrow k \leq [b_1 = w_1, \ldots, b_n = w_n, \\
&\qquad\quad \text{not } c_1 = w_{n+1}, \ldots, \text{not } c_m = w_{n+m}].
\end{aligned}
$$

- A model is supported [Apt et al., 1988] iff $M = \mathrm{T}_{PM}(M)$ and stable [Gelfond and Lifschitz, ICLP 1988] iff $M = \mathrm{LM}(P^M)$.

Standard Logic Programs

- Logic programs consist of rules of the following forms:

$$
\begin{array}{rcl}
a & \leftarrow & b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m. \\
\{a\} & \leftarrow & b_1, \ldots, b_n, \text{not } c_1, \ldots, \text{not } c_m. \\
a & \leftarrow & k \leq [b_1 = w_1, \ldots, b_n = w_n, \\
 & & \quad \text{not } c_1 = w_{n+1}, \ldots, \text{not } c_m = w_{n+m}].
\end{array}
$$

- A model is supported [Apt et al., 1988] iff $M = \mathrm{T}_{PM}(M)$ and stable [Gelfond and Lifschitz, ICLP 1988] iff $M = \mathrm{LM}(P^M)$.

Example

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

$\Longrightarrow$ $M_1 = \{a, b, c\}$ and $M_2 = \{a, b, d\}$ are both supported, and M_1 is also stable.

Acyclicity Extension

An acyclicity extension is a pair $\langle V, e\rangle$ where

1. V is a set of vertices and
2. $e : \mathrm{At}(P) \rightarrow V \times V$ is a partial injection that maps atoms of a logic program P to edges.

Acyclicity Extension

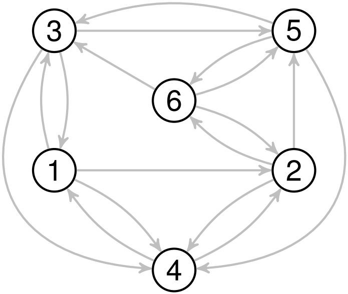
An acyclicity extension is a pair $\langle V, e \rangle$ where

1. V is a set of vertices and
2. $e : \mathrm{At}(P) \to V \times V$ is a partial injection that maps atoms of a logic program P to edges.

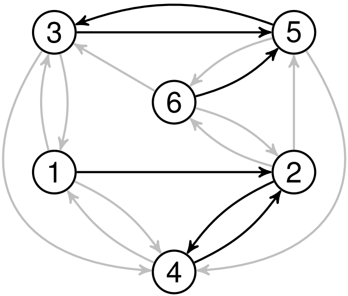
An interpretation $M \subseteq \mathrm{At}(P)$ is a stable/supported model of P subject to an acyclicity extension $\langle V, e \rangle$, iff

1. M is a stable/supported model of P and
2. the graph $\langle V, e(M) \rangle$ is acyclic, where $e(M) = \{\langle v, u \rangle \in V \times V \mid a \in M, e(a) = \langle v, u \rangle\}$.

Hamiltonian Cycles in ASP

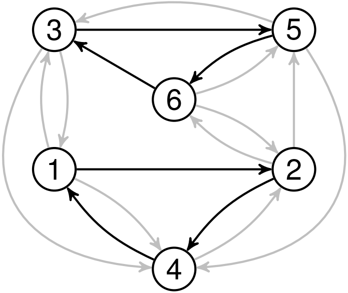


Hamiltonian Cycles in ASP



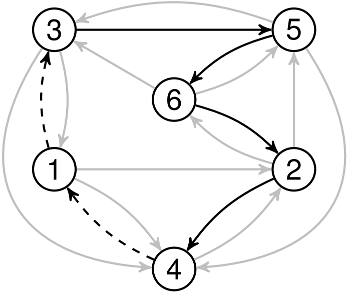

```
1 { hc(X,Y) : edge(X,Y) } 1 :- node(X).
```

Hamiltonian Cycles in ASP




```
1 { hc(X,Y) : edge(X,Y) } 1 :- node(X).
1 { hc(X,Y) : edge(X,Y) } 1 :- node(Y).
```

Hamiltonian Cycles in ASP




```
1 { hc(X,Y) : edge(X,Y) } 1 :- node(X).
1 { hc(X,Y) : edge(X,Y) } 1 :- node(Y).
_edge(X,Y) :- hc(X,Y), X > 1, Y > 1.
```

Example: Acyclicity Constraints

Let us consider a standard logic program

$$a \leftarrow b. \quad a \leftarrow c. \quad b \leftarrow a. \quad c \leftarrow \text{not } d. \quad d \leftarrow \text{not } c.$$
$$\text{\_edge}(a, b) \leftarrow a, \text{not } c. \quad \text{\_edge}(b, a) \leftarrow b.$$

and extend it by $\langle V, e \rangle$ where $V = \{a, b\}$ and e is the mapping

$$\text{\_edge}(a, b) \mapsto \langle a, b \rangle, \qquad \text{\_edge}(b, a) \mapsto \langle b, a \rangle.$$

Example: Acyclicity Constraints

Let us consider a standard logic program

$$a \leftarrow b. \quad a \leftarrow c. \quad b \leftarrow a. \quad c \leftarrow \text{not } d. \quad d \leftarrow \text{not } c.$$
$$\_\text{edge}(a, b) \leftarrow a, \text{not } c. \quad \_\text{edge}(b, a) \leftarrow b.$$

and extend it by $\langle V, e \rangle$ where $V = \{a, b\}$ and e is the mapping

$$\_\text{edge}(a, b) \mapsto \langle a, b \rangle, \qquad \_\text{edge}(b, a) \mapsto \langle b, a \rangle.$$

$\Longrightarrow$ $M_1 = \{a, b, c, \_\text{edge}(b, a)\}$ is a stable and supported model;
$M_2 = \{a, b, d, \_\text{edge}(a, b), \_\text{edge}(b, a)\}$ is neither.

Translation from ASP to ACYC-ASP

- We define a translation $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ that extends P by an acyclicity extension and a set of rules.

Translation from ASP to ACYC-ASP

- We define a translation $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ that extends P by an acyclicity extension and a set of rules.
- The stable models of P coincide with the stable/supported models of $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ modulo acyclicity.

Translation from ASP to ACYC-ASP

- We define a translation $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ that extends P by an acyclicity extension and a set of rules.
- The stable models of P coincide with the stable/supported models of $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ modulo acyclicity.
- Well-support of answer sets can be addressed by performing on $\mathrm{Tr}_{\mathrm{ACYC}}(P)$ one or both of
  - unfounded set checking or
  - acyclicity checking.

Tool Support

gringo		
lp2acyc		
lp2sat [-g]	acyc2solver [--diff] [--bv] [--pb] [--mip]	clasp --enable-acyc

These tools are published under:

http://research.ics.aalto.fi/software/asp/lp2acyc/
http://potassco.sourceforge.net/projects/potassco/

Experiments: Decision Problems

Mode	*Cycle #60*		*Laby #20*		*Soko #30*		*Route #23*	
UFS	**36.0**	**0**	255.3	**4**	182.6	**2**	**5.8**	**0**
ACYC	373.6	37	261.0	6	350.7	10	134.5	4
BCYC	266.3	26	286.7	7	256.2	7	111.5	2
ACYC/UFS	209.4	18	279.2	**4**	174.6	3	11.4	**0**
BCYC/UFS	209.2	19	314.3	6	179.7	4	10.0	**0**
ACYC+	118.0	7	366.7	7	336.7	10	137.2	4
BCYC+	85.3	5	279.6	5	230.4	5	138.6	4
ACYC+/UFS	115.9	8	311.8	5	176.6	4	15.4	**0**
BCYC+/UFS	91.9	6	**212.7**	**4**	**170.2**	3	12.3	**0**

ACYC: Acyclicity checking

BCYC: ACYC with backward propagation

UFS: Unfounded set checking

+: Extended translation

Experiments: Optimization Problems

Mode	*Bayes #30*		*Markov #21*		*Sched #18*	
UFS	116.8	**0**	100.7	**0**	**281.2**	**7**
ACYC	**66.3**	**0**	120.3	1	320.9	8
BCYC	84.6	**0**	54.1	**0**	324.2	**7**
ACYC/UFS	103.1	1	170.2	3	348.2	9
BCYC/UFS	104.3	1	72.5	**0**	340.3	9
ACYC+	106.2	1	61.5	**0**	340.9	9
BCYC+	102.2	2	**39.9**	**0**	341.1	9
ACYC+/UFS	110.3	1	171.4	3	367.5	9
BCYC+/UFS	122.5	2	111.5	1	360.6	9

ACYC: Acyclicity checking

BCYC: ACYC with backward propagation

UFS: Unfounded set checking

+: Extended translation

Conclusion

- We propose ASP modulo Acyclicity
  - to help in application areas involving DAGs, trees, etc., and
  - to embed ASP into itself.
- Well-support of answer sets can be addressed by acyclicity checking
- Implementation is built into the tools `lp2acyc` and `clasp`