Computing Answer Sets of a Logic Program via enumeration of SAT certificates

Yuliya Lierler[1] **and Marco Maratea**[2]

yuliya@cs.utexas.edu

marco@dist.unige.it

[1]University of Texas at Austin

[2]MRG-LAB DIST, Università di Genova

CMODELS - **ASP tool**

CMODELS is an answer set programming system, which computes answer sets of a logic program using SAT solver.

CMODELS first finds the completion of a program and then applies SAT solver for finding the models of completion.

Depending on tightness property of a program the completion may be extended and SAT solver may be invoked several times into the computation.

Earlier work:

- SMODELS
- LPARSE

Accepted syntax

CMODELS accepts programs that include the following rules:

Basic rules $a \leftarrow a_1, \ldots, a_n,$ **not** $a_{n+1}, \ldots,$ **not** a_m

Choice rules $\{a, \ldots, a_k\} \leftarrow a_{k+1}, \ldots, a_m,$ **not** $a_{m+1}, \ldots,$ **not** a_n

Weight constraint rules $a \leftarrow L\{a_1 = w_1, \ldots, a_m = w_m,$
not $a_{m+1} = w_{m+1}, \ldots,$ **not** $a_n = w_n\}U$

Basic nested rules

A basic nested rule is an expression of the form:

$$a \leftarrow a_1, \ldots, a_n, \textbf{not } a_{n+1}, \ldots, \textbf{not } a_m, \textbf{not not } a_{m+1}, \ldots, \textbf{not not } a_k.$$

A *basic nested program* is a finite set of basic nested rules.

Translating an input program into a basic nested program

Choice rules are translated into basic nested rules.

Weight constraint rules are eliminated by introducing auxiliary atoms, Ferraris and Lifschitz (2003).

The concept of completion for nested programs is defined by Lloyd and Topor (1984).

Completion

Given a basic nested program Π , for every atom a make the list of all rules in Π with the head a:

$$a \leftarrow Body_i$$

and form the equivalence:

$$a \equiv \bigvee_i Body_i \tag{1}$$

and, for every constraint $\bot \leftarrow Body$ in Π form the negation of its body:

$$\neg Body \tag{2}$$

The *completion Comp*(Π) of Π consists of all formulas (1) and (2).

Tight logic programs

Tightness is a syntactic condition on a logic program.

To verify tightness of a program Π we build the *positive dependency graph* G which corresponds to Π. Each atom of Π is a vertex of G and for each rule $a \leftarrow \ \ldots, b, \ldots$ in Π there is an edge between a and b in G.

The program Π is *absolutely tight* iff there are no cycles in the graph of Π.

Nontigth logic programs

Nontigth logic programs are programs in which there are *cycles* in the correspondent graphs.

Cycles (or loops) cause the difference between answer sets of a program and models of program's completion.

Example:
$p \leftarrow p$ is not tight; it has two model of completion $\emptyset$ and $\{\mathrm{p}\}$ and only one answer set $\emptyset$.

Lin and Zhao proposed in 2002, with their solver ASSAT, a method for using SAT solvers to find answer sets of nontight logic programs.

The main idea is: adding "loop formulas" to the completion eliminates "bad" models of completion.

Loop formulas

CMODELS adopts the idea of a loop formula for the case of nontight programs.

It uses extended definition of a loop formula proposed by Lee and Lifschitz (2003), in order to cover basic nested rules.

The algorithm used for finding the loop formula is very similar to the one presented by Lin and Zhao (2002).

ASSAT's algorithm

Given a logic program Π:

1. Let T be Comp(Π)
2. Find a model M of T. If there is no such model, then terminate with failure
3. If M is an answer set, then exit with it
4. If M is not an answer set, then find a loop L such that its loop formula ϕ_L is not satisfied by M
5. Let T be T U $\{\phi_L\}$ and go back to step 2

ASSAT's disadvantages

The system ASSAT has some disadvantages:

- covers only basic rules
- finds only one answer set
- may explore the same parts of the search tree already explored
- may blow up in the number of propositional clauses

CMODELS-2

CMODELS-1 (Babovich and Lifschitz 2003), is a system that does not suffer of the ASSAT's first and second disadvantages, but is limited to tight programs.

CMODELS-2 (Babovich and Maratea 2003), combines the attractive features of ASSAT and CMODELS-1.

CMODELS-2's algorithm is:

1. Let T be Comp(Π)
2. Find a model M of T. If there is no such model, then terminate with failure
3. If M is an answer set, add to T the negation of the propositional model and go back to step 2
4. If M is not an answer set, consider M as failure, then find a loop L such that its loop formula ϕ_L is not satisfied by M , find one reason from ϕ_L, backjump in the search tree with it, and go back to step 2

Experiments with CMODELS-2 (1)

Instance name	CMODELS SIMO (-bj)	CMODELS SIMO (-le)	SMODELS	CMODELS SIMO assat alg.
np40c	(42) 1.59	(42) 1.56	2.49	(27) 16.52
np60c	(115) 9.35	(106) 8.80	21.45	(35) 76.12
np70c	(172) 20.75	(217) 26.46	42.86	(41) 139.50
np80c	(278) 43.92	(223) 37.87	79.78	(44) 241.55
np100c	(406) 103.22	(286) 78.93	200.43	(51) 561.94
np120c	—	(698) 314.93	430.98	mem
np150c	—	(1074) 841.91	1171.38	mem

Figura 1: Complete graphs CMODELS employing backjumping (and learning) vs. SMODELS vs. CMODELS employing assat algorithm.

Experiments with CMODELS-2 (2)

Instance name	CMODELS SIMO (-bj)	CMODELS SIMO (-le)	SMODELS	CMODELS SIMO assat alg.
2xp30	(0) 0.01	(0) 0.01	0.01	(0) 0.01
2xp30.1	timeout	timeout	0.12	(90) 57.58
2xp30.2	timeout	(155) 3092.13	timeout	(152) 24.12
2xp30.4	timeout	timeout	timeout	timeout
4xp20	(0) 0.01	(0) 0.01	0.01	(0) 0.01
4xp20.1	(23) 61.88	(2) 73.26	timeout	(1) 2.03
4xp20.3	(43) 89.37	(13) 82.90	0.01	(5) 1.56

Figura 2: Hand-coded graphs CMODELS employing backjumping (and learning) vs. SMODELS vs. CMODELS employing assat algorithm.

Experiments with CMODELS-2 (3)

Instance Name	CMODELS MCHAFF	CMODELS SIMO (-bj)	CMODELS SIMO (-le)	SMODELS
dp_6.formula1-i-O2-b8	(28) 6.17	(6) 0.45	(6) 0.43	0.46
dp_8.formula1-s-O2-b8	(15) 5.14	(29) 1.02	(14) 0.94	1.83
dp_8.formula1-i-O2-b10	(24) 28.72	(55) 7.22	(41) 6.61	5.08
dp_10.formula1-s-O2-b9	(24) 19.47	(89) 4.17	(36) 3.51	29.05
dp_10.formula1-i-O2-b12	(21) 51.36	(719) 73.09	(162) 14.34	428.85
dp_12.formula1-s-O2-b10	(69) 96.95	(100) 8.49	(93) 7.56	949.95
dp_12.formula1-i-O2-b14	(29) 469.83	(24) 242.56	(14) 81.80	timeout

Figura 3: Nontight Bounded Model Checking CMODELS using MCHAFF employing ASSAT-like algorithm and SIMO employing backjumping (and learning) vs. SMODELS

Final question

Will we see a Version 3 of CMODELS that can compete with DLV?