Representing Normal Programs with Clauses

Tomi Janhunen

Tomi.Janhunen@hut.fi

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science

August 26, 2004

MOTIVATION

- Our goal is to combine the knowledge representation capabilities of normal logic programs with the efficiency of SAT solvers.

$$\begin{array}{ccc}
\boxed{\text{Normal program } P} & & \text{answer set} \\
\downarrow & & \\
\boxed{\text{Set of clauses } S} & & \uparrow \\
\downarrow & & \\
\boxed{\text{SAT solver}} & \longrightarrow & \text{model}
\end{array}$$

- To realize this setting, we present a polynomial and faithful but non-modular translation.

OUTLINE

① Terms and Definitions

② Characterizing Stability

③ Clausal Representation

④ Experiments

⑤ Discussion

① TERMS AND DEFINITIONS

- A rule r is an expression of the form

$$\mathsf{h} \leftarrow \mathsf{b}_1, \ldots, \mathsf{b}_n, {\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m.$$

- We use the following notations for a rule r:

$$\begin{aligned}
\mathrm{H}(r) &= \mathsf{h} && \text{(head)} \\
\mathrm{B}(r) &= \{\mathsf{b}_1, \ldots, \mathsf{b}_n, {\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m\} && \text{(body)} \\
\mathrm{B}^+(r) &= \{\mathsf{b}_1, \ldots, \mathsf{b}_n\} \\
\mathrm{B}^-(r) &= \{\mathsf{c}_1, \ldots, \mathsf{c}_m\}
\end{aligned}$$

- We define normal logic programs, or normal programs for short, as sets of rules.

Syntactic Restrictions

- We distinguish the following special cases:
  — positive rules: $\mathsf{h} \leftarrow \mathsf{b}_1, \ldots, \mathsf{b}_n$
  — atomic rules: $\mathsf{h} \leftarrow {\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m$
  — strictly unary rules: $\mathsf{h} \leftarrow \mathsf{b}, {\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m$
  — strictly binary rules: $\mathsf{h} \leftarrow \mathsf{b}_1, \mathsf{b}_2, {\sim}\mathsf{c}_1, \ldots, {\sim}\mathsf{c}_m$
- We extend these conditions for sets of rules:

  — positive programs: $\forall r \in P : |\mathrm{B}^-(r)| = 0$
  — atomic programs: $\forall r \in P : |\mathrm{B}^+(r)| = 0$
  — unary programs: $\forall r \in P : |\mathrm{B}^+(r)| \leq 1$
  — binary programs: $\forall r \in P : |\mathrm{B}^+(r)| \leq 2$

Least Models

If P is a positive normal program, then

1. P has a unique minimal model, i.e. the least model $\mathrm{LM}(P)$ of P;
2. $\mathrm{LM}(P) = \mathrm{T}_P \uparrow \omega = \mathrm{lfp}(\mathrm{T}_P)$ where the immediately true operator T_P is defined by

   $$\mathrm{T}_P(A) = \{\mathrm{H}(r) \mid r \in P \text{ and } \mathrm{B}^+(r) \subseteq A\};$$

   and
3. $\mathrm{lfp}(\mathrm{T}_P) = \mathrm{T}_P \uparrow i$ for some $i \in \mathbb{N}$, if P is finite.

Level Numbers

Definition: For each atom $\mathsf{b} \in \mathrm{LM}(P)$, the level number $\mathrm{lev}(\mathsf{b})$ of b is the least number n such that $\mathsf{b} \in \mathrm{T}_P \uparrow n - \mathrm{T}_P \uparrow (n-1)$.

Example: Consider a positive normal program

$$P = \{r_1 = \mathsf{a} \leftarrow;\quad r_2 = \mathsf{a} \leftarrow \mathsf{b};\quad r_3 = \mathsf{b} \leftarrow \mathsf{a}\}$$

with $\mathrm{LM}(P) = \{\mathsf{a}, \mathsf{b}\}$ and the corresponding level numbers $\mathrm{lev}(\mathsf{a}) = 1$ and $\mathrm{lev}(\mathsf{b}) = 2$.

Stable and Supported Models

Definition: Given an interpretation M, the Gelfond-Lifschitz reduct

$$P^M = \{r^+ \mid r \in P \text{ and } \mathrm{B}^-(r) \cap M = \emptyset\}$$

where r^+ is defined as $\mathrm{H}(r) \leftarrow \mathrm{B}^+(r)$ for $r \in P$.

Definition: For a normal program P, an interpretation $M \subseteq \mathrm{At}(P)$ is

1. a stable model of $P \iff M = \mathrm{LM}(P^M)$, and
2. a supported model of $P \iff M = \mathrm{T}_{P^M}(M)$.

Stable and Supported Models

Example: The normal program
$P = \{\mathsf{a} \leftarrow \mathsf{b};\ \ \mathsf{b} \leftarrow \mathsf{a}\}$ has two supported models
$M_1 = \emptyset$ and $M_2 = \{\mathsf{a}, \mathsf{b}\}$, but only M_1 is stable, as
$\mathrm{LM}(P^{M_1}) = \mathrm{LM}(P) = \emptyset = M_1$ and
$\mathrm{LM}(P^{M_2}) = \mathrm{LM}(P) = \emptyset \neq M_2$.

Some important properties:

1. Stable models are also supported models.
2. Stable and supported models coincide for atomic programs.
3. Clark's completion captures supported models.

② CHARACTERIZING STABILITY

Definition: Let M be a supported model of a normal program P. A level numbering w.r.t. M is a function $\# : M \cup \mathrm{SR}(P, M) \to \mathbb{N}$ such that

1. for all $\mathsf{a} \in M$,
   $\#\mathsf{a} = \min\{\#r \mid r \in \mathrm{SR}(P, M) \text{ and } \mathsf{a} = \mathrm{H}(r)\}$ and
2. for all $r \in \mathrm{SR}(P, M)$,
   $\#r = \max\{\#\mathsf{b} \mid \mathsf{b} \in \mathrm{B}^+(r)\} + 1$

where $\mathrm{SR}(P, M) = \{r \in P \mid M \models \mathrm{B}(r)\}$.

We define $\max \emptyset = 0$ to cover rules r with $\mathrm{B}^+(r) = \emptyset$.

Capturing Stable Models

Let M be a supported model of P.

Proposition: If $\#$ is a level numbering w.r.t. M, then it is unique.

Theorem: M is a stable model of P
$\iff$ there is a level numbering $\#$ w.r.t. M.

Capturing Stable Models

Example: Recall the supported models of
$P = \{r_1, r_2\}$ with $r_1 = \mathsf{a} \leftarrow \mathsf{b}$ and $r_2 = \mathsf{b} \leftarrow \mathsf{a}$:
$M_1 = \emptyset$ and $M_2 = \{\mathsf{a}, \mathsf{b}\}$.

- Since $M_1 \cup \mathrm{SR}(P, M_1) = \emptyset$, M_1 is trivially stable.
- For M_2, the domain $M_2 \cup \mathrm{SR}(P, M_2) = M_2 \cup P$ and the set of equations

$$\#\mathsf{a} = \#r_1,\ \#r_1 = \#\mathsf{b} + 1,$$
$$\#\mathsf{b} = \#r_2,\ \#r_2 = \#\mathsf{a} + 1$$

has no solution. Thus M_2 is not stable.

③ CLAUSAL REPRESENTATION

- We use an atomic normal program $\mathrm{Tr}_{\mathrm{AT}}(P) =$

  $$\mathrm{Tr}_{\mathrm{SUPP}}(P) \cup \mathrm{Tr}_{\mathrm{CTR}}(P) \cup \mathrm{Tr}_{\mathrm{MIN}}(P) \cup \mathrm{Tr}_{\mathrm{MAX}}(P)$$

  as an intermediary representation when translating a normal program P into a set of clauses $\mathrm{Tr}_{\mathrm{CL}}(\mathrm{Tr}_{\mathrm{AT}}(P))$.
- Level numbers have to be captured using binary counters which are represented by vectors of propositional atoms.
- Certain primitives have to be represented: $\mathrm{SEL}(c)$, $\mathrm{NXT}(c,d)$, $\mathrm{FIX}(c)$, $\mathrm{LT}(c,d)$, $\mathrm{EQ}(c,d)$.

Optimizations

- The level numbers associated with rules can be totally omitted, if all non-binary rules r with $|\mathrm{B}^{+}(r)| > 2$ are translated away.
- A normal logic program P is partitioned into its strongly connected components $C_1, \ldots, C_n$ on the basis of positive dependencies.
- No counters are needed, if $|\mathrm{H}(C_i)| = 1$ holds.
- The number of bits $\nabla C_i = \lceil \log_2(|\mathrm{H}(C_i)| + 2) \rceil$ for other strongly connected components C_i.
- Fixed translation schemes can be devised for atomic, strictly unary, and strictly binary rules.

Example

For $P = \{\mathsf{a} \leftarrow \mathsf{b};\ \ \mathsf{b} \leftarrow \mathsf{a}\}$, the translation $\mathrm{Tr}_{\mathrm{AT}}(P)$ contains the following rules for a:

$$\mathsf{b} \leftarrow {\sim}\overline{\mathsf{bt}(r_2)};\quad \overline{\mathsf{bt}(r_2)} \leftarrow {\sim}\mathsf{bt}(r_2);\quad \mathsf{bt}(r_2) \leftarrow {\sim}\overline{\mathsf{a}};$$
$$\overline{\mathsf{a}} \leftarrow {\sim}\mathsf{a};\quad \mathsf{x} \leftarrow {\sim}\mathsf{x}, {\sim}\overline{\mathsf{a}}, {\sim}\mathsf{min}(\mathsf{a});$$
$$\mathsf{x} \leftarrow {\sim}\mathsf{x}, {\sim}\overline{\mathsf{bt}(r_2)}, {\sim}\overline{\mathsf{lt}(\mathsf{nxt}(\mathsf{a}), \mathsf{ctr}(\mathsf{b}))_1};\quad \text{and}$$
$$\mathsf{min}(\mathsf{b}) \leftarrow {\sim}\overline{\mathsf{bt}(r_2)}, {\sim}\overline{\mathsf{eq}(\mathsf{nxt}(\mathsf{a}), \mathsf{ctr}(\mathsf{b}))}$$

in addition to four subprograms for choosing the values of $\mathsf{ctr}(\mathsf{a})$ and $\mathsf{nxt}(\mathsf{a})$ as well as comparing the latter with $\mathsf{ctr}(\mathsf{b})$. The rules for b are symmetric.

The only stable model is $N = \{\overline{\mathsf{a}}, \overline{\mathsf{b}}, \overline{\mathsf{bt}(r_1)}, \overline{\mathsf{bt}(r_2)}\}$.

④ EXPERIMENTS

- We have implemented $\mathrm{Tr}_{\mathrm{AT}}$ and $\mathrm{Tr}_{\mathrm{CL}}$ as respective translators LP2ATOMIC and LP2SAT to be used together with LPARSE.
- Our experiments were run on a 1.67 GHz CPU with 1GB memory.
- In our benchmark, we compute all subgraphs of D_n whose all vertices are mutually reachable.

Here D_n is a directed graph with n vertices and $n^2 - n$ edges: $E_n = \{\langle i,j \rangle \mid 0 < i \leq n,\ 0 < j \leq n,\ i \neq j\}$.

Reachability Benchmark

```
vertex(1..n).
in(V1,V2) :- not out(V1,V2),
             vertex(V1;V2), V1!=V2.
out(V1,V2) :- not in(V1,V2),
              vertex(V1;V2), V1!=V2.
reach(V,V) :- vertex(V).
reach(V1,V3) :- in(V1,V2), reach(V2,V3),
                vertex(V1;V2;V3), V1!=V2, V1!=V3.
:- not reach(V1,V2), vertex(V1;V2).
```

☞ The order in which the reachability of nodes inferred cannot be determined beforehand.

Computing All Solutions

Number of Vertices	1	2	3	4	5
SMODELS	0.004	0.003	0.003	0.033	12
CMODELS	0.031	0.030	0.124	293	-
LP2ATOMIC+SMODELS	0.004	0.008	0.013	0.393	353
LP2SAT+CHAFF	0.011	0.009	0.023	1.670	-
LP2SAT+RELSAT	0.004	0.005	0.018	0.657	1879
WF+LP2SAT+RELSAT	0.009	0.013	0.018	0.562	1598
Models	1	1	18	1606	565080
SCCs with $\lvert \mathrm{H}(C) \rvert > 1$	0	0	3	4	5
Rules (LPARSE)	3	14	39	84	155
Rules (LP2ATOMIC)	3	18	240	664	1920
Clauses (LP2SAT)	4	36	818	2386	7642
Clauses (WF+LP2SAT)	2	10	553	1677	5971

Computing Only One Solution

Number of Vertices	8	9	10
SMODELS	0.009	0.013	0.022
CMODELS	0.046	0.042	0.055
LP2ATOMIC+SMODELS	$>10^4$	$>10^4$	$>10^4$
LP2SAT+CHAFF	0.771	32.6	254
LP2SAT+RELSAT	2.51	$>10^4$	$>10^4$
WF+LP2SAT+RELSAT	2.80	4830	$>10^4$
ASSAT	0.023	0.028	0.037

⑤ DISCUSSION

- The new characterization of stable models is based on canonical level numberings.
- The translation function $\mathrm{Tr}_{\mathrm{AT}} \circ \mathrm{Tr}_{\mathrm{CL}}$ has distinctive properties:
  - it covers all finite normal programs P,
  - a bijective relationship of models is obtained,
  - the Herbrand base $\mathrm{At}(P)$ is preserved,
  - the length $\lvert\lvert \mathrm{Tr}_{\mathrm{CL}}(\mathrm{Tr}_{\mathrm{AT}}(P)) \rvert\rvert$ is of order $\lvert\lvert P \rvert\rvert \times \log_2 \lvert \mathrm{At}(P) \rvert$, and
  - incremental updating is not needed.

Conclusions and Future Work

- Various kinds of closures of relations, such as transitive closure, can be properly captured with classical models.
- Our approach is competitive against other SAT-solver-based approaches when the task is to compute all stable models.
- Further optimizations should be pursued for in order to really compete with SMODELS.
- In the future, we intend to study techniques to reduce the number of binary counters and the numbers of bits involved in them.