Systems and Solving Techniques for Knowledge Representation

– (Normal) ASP solving [Part II - ASP] –

Marco Maratea
University of Genoa, Italy

066 011 Double degree programme Computational Logic
066 931 Computational Intelligence
066 937 Software Engineering & Internet Computing
Institute of Information Systems

Abstract CMODELS with backtracking

Given a logic program Π, consider

- the (plain) CNF conversion of the completion $Comp(\Pi)$ which consists, for every $a \in atoms(\Pi)$, of clauses:
  1. the rules $a \leftarrow B$ of Π written as clauses
  $$a \vee \overline{B}$$
  2. formulas
  $$\overline{a} \vee \bigvee_{B \in Bodies(\Pi,a)} B$$
  converted to CNF using the distributivity of disjunction over conjunction (repetitions not removed)
- the conjunction of all loop formulas of Π, $LF(\Pi)$, where given a loop L, we define $R(L, a)$ to be the set of formulas
  $$b_1 \wedge \cdots \wedge b_l \wedge \overline{b_{l+1}} \wedge \cdots \wedge \overline{b_m}$$
  for all rules in Π, with $a \in L$ and $\{b_1, \ldots b_k\} \cap L = \emptyset$. The loop formula associated with L is
  $$\vee_{p \in L} l \rightarrow \vee_{a \in L} R(L, a)$$

Abstract CMODELS with backtracking: Example

Given the following program Π,

$$a \leftarrow a.$$

Initial state :		$\emptyset$
Decide	$\Longrightarrow$	a^{Δ}
Test	$\Longrightarrow$	$a^{\Delta}\overline{a}$
Backtrack	$\Longrightarrow$	$\overline{a}$
Success	$\Longrightarrow$	$Ok(\overline{a})$

Figure : Example of path in $GT_{\{a \vee \overline{a}, \overline{a}\}}$.

$\{\overline{a}\}^{+} = \emptyset$ is an (the only) answer set of Π.

G&T with learning: Extended GT state and graph

For a CNF formula F, and a formula G formed from atoms $atoms(F)$, an *extended (GT) state* relative to F and G is either

1. a pair (L, Γ), written $L||\Gamma$, where
   - L is a record relative to $atoms(F)$, and
   - Γ is a set of clauses over $atoms(F)$ that are entailed by $F \wedge G$; or
2. the distinguished state $Ok(L)$ or $UNSAT$.

$GTL_{F,G}$ graph

1. Its nodes are extended GT states relative to F and G, and
2. its transition rules are *UnitLearn*, *Decide*, *Conclude*, *Success* of $DPLLearn_F$, plus the three following rules.

G&T with learning: Extended and Updated rules

$$\textit{BackjumpGT}: \quad Ll^{\Delta}L'\|\Gamma \Longrightarrow Ll'\|\Gamma \quad \text{if} \begin{cases} Ll^{\Delta}L' \text{ is inconsistent and} \\ F \wedge G \models l' \vee \overline{L} \end{cases}$$

$$\textit{LearnGT}: \quad L\|\Gamma \Longrightarrow L\|C \cup \Gamma \quad \text{if} \begin{cases} \text{every atom in } C \text{ occurs in } F \text{ and} \\ F \wedge G \models C \end{cases}$$

$$\textit{Test}: \quad L\|\Gamma \Longrightarrow L\bar{l}\|\Gamma \quad \text{if} \begin{cases} L \text{ is consistent and} \\ G \models \overline{L} \text{ and} \\ l \in L \end{cases}$$

G&T with learning: Formal result

Theorem

For any CNF formula F and a formula G formed from atoms(F)

1. *every path in $GTL_{F,G}$ uses only finitely many times edges justified by transition rules other than Learn,*
2. *any terminal state reachable from $\emptyset$ in $GTL_{F,G}$ other than UNSAT is Ok(L), with L being a model of $F \wedge G$, and*
3. *UNSAT is reachable from $\emptyset$ in $GT_{F,G}$ if and only if $F \wedge G$ is unsatisfiable.*

Abstract CMODELS

Given a logic program Π, if

- F is the CNF conversion of the completion $Comp(\Pi)$, and
- G is $LF(\Pi)$,

Abstract CMODELS with learning

- $GTL_{Comp(\Pi),LF(\Pi)}$ abstracts CMODELS with learning [Lierler, 2005] implementing ASP-SAT procedure+learning [Giunchiglia et al., 2006], by
  1. applying *LearnGT* in a state reached by the application of *BackjumpGT*, and
  2. assigning priorities to the application of the transition rules as follows: *BackjumpGT*, *Conclude* >> *UnitLearn* >> *Decide* >> *Test*. Such ordering guarantees that
     - *Test* is applied only on models of $F \cup \Gamma$, and
     - *BackjumpGT* is first applied on a state reached by the application of *Test*.
- If the state $Ok(L)$ is reached, then L^+ is an answer set of Π.

Abstract CMODELS

Given a logic program Π, if

- F is the CNF conversion of the completion $Comp(\Pi)$, and
- G is $LF(\Pi)$,

Abstract CMODELS with learning

- $GTL_{Comp(\Pi),LF(\Pi)}$ abstracts CMODELS with learning [Lierler, 2005] implementing ASP-SAT procedure+learning [Giunchiglia et al., 2006], by
  1. applying *LearnGT* in a state reached by the application of *BackjumpGT*, and
  2. assigning priorities to the application of the transition rules as follows: *BackjumpGT*, *Conclude* $>>$ *UnitLearn* $>>$ *Decide* $>>$ *Test*. Such ordering guarantees that
     - *Test* is applied only on models of $F \cup \Gamma$, and
     - *BackjumpGT* is first applied on a state reached by the application of *Test*.
- If the state $Ok(L)$ is reached, then L^+ is an answer set of Π.

Abstract CLASP

CLASP [Gebser et al., 2007]

- Employs an additional rule wrt $GTL_{F,G}$:

$$\textit{Unfounded}: \quad L \Longrightarrow L\bar{a} \quad \text{if} \begin{cases} L \text{ is consistent and} \\ a \in U \text{ for a set } U \text{ unfounded on } L \text{ w.r.t. } \Pi \end{cases}$$

  A set of ground atoms U is an unfounded set if, for each rule r s.t. $H(r) \in U$, one of the following conditions hold

  1. the body of r is false w.r.t. U, or
  2. some literal in the positive body belongs to U.

- Follows the ordering on rules application: *BackjumpGT*, *Conclude* $>>$ *UnitLearn*, *Unfounded* $>>$ *Decide*.
- Applies *LearnGT* in a state reached by the application of *BackjumpGT*.

Abstract $ATLEAST_\Pi$

We now define a graph whose terminal nodes correspond to supported models of a program Π.

$ATLEAST_\Pi$ graph

1. Its nodes are the states relative to the set of atoms $atoms(\Pi)$, and
2. its edges are justified by the transition rules *Decide*, *Conclude*, *Backtrack*, *Success* of the *DPLL* graph, and some additional rules that describe deterministic consequences.

Abstract $ATLEAST_{\Pi}$

We now define a graph whose terminal nodes correspond to supported models of a program Π.

$ATLEAST_{\Pi}$ graph

1. Its nodes are the states relative to the set of atoms $atoms(\Pi)$, and
2. its edges are justified by the transition rules *Decide*, *Conclude*, *Backtrack*, *Success* of the *DPLL* graph, and some additional rules that describe deterministic consequences.

Abstract $ATLEAST_\Pi$: Additional rules

$$UnitPropagateLP: \quad L \Longrightarrow La \quad \text{if} \begin{cases} \text{there is a rule } a \leftarrow B \text{ of } \Pi \text{ such that} \\ \quad B \subseteq L \end{cases}$$

$$AllRulesCancelled: \quad L \Longrightarrow L\bar{a} \quad \text{if} \begin{cases} \text{for each rule } a \leftarrow B \text{ of } \Pi \\ \quad B \text{ is contradicted by } L \end{cases}$$

$$BackchainTrue: \quad L \Longrightarrow Ll \quad \text{if} \begin{cases} \text{there is a rule } a \leftarrow l, B \text{ of } \Pi \text{ such that} \\ \quad a \text{ is in } L \text{ and} \\ \text{for each other rule } a \leftarrow B' \text{ of } \Pi \\ \quad B' \text{ is contradicted by } L \end{cases}$$

$$BackchainFalse: \quad L \Longrightarrow L\bar{l} \quad \text{if} \begin{cases} \text{there is a rule } a \leftarrow l, B \text{ of } \Pi \text{ such that} \\ \quad \bar{a} \text{ is in } L \text{ or } a = \bot \text{ and} \\ \quad B \subseteq L \end{cases}$$

Abstract $ATLEAST_\Pi$: Formal result

Theorem

For any program Π,

1. *graph* $ATLEAST_\Pi$ *is finite and acyclic,*
2. *any terminal state reachable from* $\emptyset$ *in* $ATLEAST_\Pi$ *other than* $UNSAT$ *is* $Ok(L)$, *with* L *being a supported model of* Π, *and*
3. $UNSAT$ *is reachable from* $\emptyset$ *in* $ATLEAST_\Pi$ *if and only if* Π *has no supported models.*

Abstract $ATLEAST_\Pi$: Formal result (II)

Theorem [Lierler, 2011]

For any program Π, the graphs $ATLEAST_\Pi$ and $DPLL_{Comp(\Pi)}$ are equal.

Abstract $ATLEAST_{\Pi}$: Example

Let Π be the following program:

$$\begin{aligned} a &\leftarrow not\ b. \\ b &\leftarrow not\ a. \\ c &\leftarrow a. \\ d &\leftarrow d. \end{aligned}$$

Initial state :		$\emptyset$
Decide	$\Longrightarrow$	a^{Δ}
UnitPropagateLP	$\Longrightarrow$	$a^{\Delta} c$
AllRulesCancelled	$\Longrightarrow$	$a^{\Delta} c \overline{b}$
Decide	$\Longrightarrow$	$a^{\Delta} c \overline{b} d^{\Delta}$
Success	$\Longrightarrow$	$Ok(a^{\Delta} c \overline{b} d^{\Delta})$

$\{a, c, \overline{b}, d\}$ *is a supported model*

Figure : Example of path in $ATLEAST_{\Pi}$.

Abstract SMODELS: SM_{Π} Graph

SM_{Π} graph

- Its nodes are the same as of the graph $ATLEAST_{\Pi}$, and
- its edges are justified by the transition rules of $ATLEAST_{\Pi}$ and *Unfounded*

$$Unfounded: \qquad L \Longrightarrow L\overline{a} \quad \text{if} \begin{cases} L \text{ is consistent and} \\ a \in U \text{ for a set } U \text{ unfounded on } L \text{ w.r.t. } \Pi \end{cases}$$

SM_Π: Formal result

Theorem

For any program Π,

1. *graph SM_Π is finite and acyclic,*
2. *any terminal state reachable from $\emptyset$ in SM_Π other than UNSAT is $Ok(L)$, with L^+ being an answer set of Π, and*
3. *UNSAT is reachable from $\emptyset$ in SM_Π if and only if Π has no answer sets.*

SM_Π: Example

Let Π be the following program:

$$a \leftarrow \textit{not } b.$$
$$b \leftarrow \textit{not } a.$$
$$c \leftarrow a.$$
$$d \leftarrow d.$$

Initial state :		$\emptyset$
Decide	$\Longrightarrow$	a^Δ
UnitPropagateLP	$\Longrightarrow$	$a^\Delta c$
AllRulesCancelled	$\Longrightarrow$	$a^\Delta c\overline{b}$
Decide	$\Longrightarrow$	$a^\Delta c\overline{b}d^\Delta$
...	...	...

$\{a, c, \overline{b}, d\}$ is a supported model of Π, but not an answer set.

SM_Π: Example (II)

Let Π be the following program:

$$\begin{array}{c} a \leftarrow not\ b. \\ b \leftarrow not\ a. \\ c \leftarrow a. \\ d \leftarrow d. \end{array}$$

$$\begin{array}{lcl}
\text{Initial state :} & & \emptyset \\
Decide & \Longrightarrow & a^\Delta \\
UnitPropagateLP & \Longrightarrow & a^\Delta c \\
AllRulesCancelled & \Longrightarrow & a^\Delta c \overline{b} \\
Decide & \Longrightarrow & a^\Delta c \overline{b} d^\Delta \\
Unfounded & \Longrightarrow & a^\Delta c \overline{b} d^\Delta \overline{d} \\
Backtrack & \Longrightarrow & a^\Delta c \overline{b}\, \overline{d} \\
Success & \Longrightarrow & Ok(a^\Delta c \overline{b}\, \overline{d})
\end{array}$$

Figure : Example of path in SM_Π.

Abstract SMODELS via SM_Π

SMODELS [Simons et al., 2002] priorities

Backtrack, *Conclude* >>
UnitPropagateLP, *AllRulesCancelled*, *BackchainTrue*, *BackchainFalse* >>
Unfounded >> *Decide*.

Initial state :		$\emptyset$
Decide	$\Longrightarrow$	a^Δ
UnitPropagateLP	$\Longrightarrow$	$a^\Delta c$
AllRulesCancelled	$\Longrightarrow$	$a^\Delta c\overline{b}$
Unfounded	$\Longrightarrow$	$a^\Delta c\overline{b}\,\overline{d}$
Success	$\Longrightarrow$	$Ok(a^\Delta c\overline{b}\,\overline{d})$

Figure : Example of path followed by SMODELS on Π.

Abstract SMODELS$_{cc}$: Graph SML_Π (Idea)

[Ward and Schlipf, 2004]

For a program Π, an *extended state* relative to Π is either

1. a pair (L, Γ), written $L \| \Gamma$, where
   - L is a record relative to $atoms(\Pi)$, and
   - Γ is a set of constraints over $atoms(\Pi)$ that are entailed by Π; or
2. the distinguished state $Ok(L)$ or $UNSAT$.

SML_Π graph

- Its nodes are the extended states relative to Π, and
- its edges are justified by extended, updated and additional transition rules wrt SM_Π.

SUP: A new solver with few changes [Lierler, 2008]

The graph of SUP_Π is a subgraph of SM_Π with

- the same nodes, and
- the same transition rules but *Unfounded*, which is now

$$\mathit{UnfoundedSUP}: \quad L||\Gamma \Longrightarrow L\bar{a}||\Gamma \quad \text{if} \begin{cases} \text{no atom is unassigned by } L \\ L \text{ is consistent and} \\ a \in U \text{ for a set } U \text{ unfounded on } L \end{cases}$$

In [Lierler, 2011] SUP_Π has been extended with backjumping and learning rules of SML_Π, with the following priorities:
BackjumpLP, *Conclude* >>
UnitPropagateLP, *AllRulesCancelled*, *BackchainTrue*, *BackchainFalse* >>
Decide >> *Unfounded*.

The implementation of SUP led to positive results. SUP participated to the 3rd ASP Competition.

SUP: A new solver with few changes [Lierler, 2008]

The graph of SUP_Π is a subgraph of SM_Π with

- the same nodes, and
- the same transition rules but *Unfounded*, which is now

$$UnfoundedSUP : \quad L||\Gamma \Longrightarrow L\overline{a}||\Gamma \quad \text{if} \begin{cases} \text{no atom is unassigned by } L \\ L \text{ is consistent and} \\ a \in U \text{ for a set } U \text{ unfounded on } L \end{cases}$$

In [Lierler, 2011] SUP_Π has been extended with backjumping and learning rules of SML_Π, with the following priorities:
BackjumpLP, *Conclude* $>>$
UnitPropagateLP, *AllRulesCancelled*, *BackchainTrue*, *BackchainFalse* $>>$
Decide $>>$ *Unfounded*.

The implementation of SUP led to positive results. SUP participated to the 3rd ASP Competition.

SUP: A new solver with few changes [Lierler, 2008]

The graph of SUP_Π is a subgraph of SM_Π with

- the same nodes, and
- the same transition rules but *Unfounded*, which is now

$$UnfoundedSUP: \quad L||\Gamma \Longrightarrow L\overline{a}||\Gamma \quad \text{if} \begin{cases} \text{no atom is unassigned by } L \\ L \text{ is consistent and} \\ a \in U \text{ for a set } U \text{ unfounded on } L \end{cases}$$

In [Lierler, 2011] SUP_Π has been extended with backjumping and learning rules of SML_Π, with the following priorities:
BackjumpLP, *Conclude* $>>$
UnitPropagateLP, *AllRulesCancelled*, *BackchainTrue*, *BackchainFalse* $>>$
Decide $>>$ *Unfounded*.

The implementation of SUP led to positive results. SUP participated to the 3rd ASP Competition.

References I

Gebser, M., Kaufmann, B., Neumann, A., and Schaub, T. (2007).
CLASP: A conflict-driven answer set solver.
In *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*.

Giunchiglia, E., Lierler, Y., and Maratea, M. (2006).
Answer set programming based on propositional satisfiability.
Journal of Automated Reasoning, 36:345–377.

Lierler, Y. (2005).
Cmodels: SAT-based disjunctive answer set solver.
In Baral, C., Greco, G., Leone, N., and Terracina, G., editors, *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005)*, volume 3662 of *Lecture Notes in Computer Science*, pages 447–452.

References II

Lierler, Y. (2008).
Abstract answer set solvers.
In de la Banda, M. G. and Pontelli, E., editors, *Proceedings of the 24th International Conference on Logic Programming (ICLP 2008)*, volume 5366 of *Lecture Notes in Computer Science*, pages 377–391. Springer.

Lierler, Y. (2011).
Abstract answer set solvers with backjumping and learning.
Theory and Practice of Logic Programming, 11:135–169.

Simons, P., Niemelä, I., and Soininen, T. (2002).
Extending and implementing the stable model semantics.
Artificial Intelligence, 138:181–234.

Ward, J. and Schlipf, J. (2004).
Answer set programming with clause learning.
In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'04)*, pages 302–313.