Computing Loops with at Most One External Support Rule for Basic Logic Programs with Arbitrary Constraint Atoms

Jianmin Ji  Fangzhen Lin  Jia-Huai You

University of Science and Technology of China
Hong Kong University of Science and Technologies
University of Alberta, Canada

ICLP 2013
Outline

1 Motivations and Preliminaries

2 Our Work

3 Concluding Remark
1 Motivations and Preliminaries

2 Our Work

3 Concluding Remark
Motivations

- Previous work has shown that:
  - For normal and disjunctive programs, the well-founded models can be computed by unit propagation on program completion and loop formulas of loops with no external support.
  - When loop formulas of loops with exactly one external support are added, consequences beyond the well-founded model can be computed, which sometimes can significantly speed up answer set computation.
  - We extend this approach to basic logic programs with abstract constraint atoms.
An *abstract constraint atom (c-atom)* $A$ is an expression of the form $(D, C)$, where $D$ is a finite set of atoms and $C \subseteq 2^D$. Specially, $A_d = D$ and $A_c = C$.

A *basic logic program with c-atoms (logic program or program)* is a finite set of *rules* of the form

$$a \leftarrow A_1, \ldots, A_k, \text{not } A_{k+1}, \ldots, \text{not } A_n$$

where $a$ is an atom and $A_i$’s are c-atoms.

The notions of *r-answer set* and *c-answer set* are defined in (Son et al. JAIR 2007).
Loops and Loop Formulas

A compact representation of c-atoms: Let $S$ and $J$ be two disjoint sets of atoms, $S \uplus J = \{S' \mid S \subseteq S' \text{ and } S' \subseteq S \cup J\}$. $S \uplus J$ is maximal in $A$ if $S \uplus J \subseteq A_c$ and there is no other sets $S'$ and $J'$ s.t. $S' \uplus J' \subseteq A_c$ and $S \uplus J \subset S' \uplus J'$. $A_c^*$ denotes the set of all maximal $S \uplus J$ in $A$.

The dependency graph $G_P$ of a basic program $P$ is a directed graph for atoms. $(u, v)$ is a directed edge if there is a rule $r \in P$ such that $u = \text{head}(r)$ and $v \in S$, for some $S \uplus J \in A_c^*$ and $A \in \text{body}(r)$.

A set $L$ of atoms is called a loop of $P$, if $L$-induced subgraph of $G_P$ is strongly connected.
Let
\[
\pi_A(L) = \bigvee_{S \uplus J \in A^*_{c | L}} S \land \neg (A_d \setminus (S \cup J)),
\]
where \(A^*_{c | L} = \{S \uplus J \in A^*_c \mid L \cap S = \emptyset\}\),
\[
\sigma_A = \bigvee_{S \uplus J \in A^*_c} S \land \neg (A_d \setminus (S \cup J)),
\]
For a basic rule \(r\) of the form (1), we define the formula
\[
\theta_L(r) = \pi_{A_1}(L) \land \cdots \land \pi_{A_k}(L) \land \neg \sigma_{A_{k+1}} \land \cdots \land \neg \sigma_{A_n}.
\]
A rule \(r \in P\) is an external support of \(L\) if \(\text{head}(r) \in L\) and \(\theta_L(r) \not\equiv \bot\).
The loop formula for \(L\) of \(P\), denoted \(LF_P(L)\), is defined as
\[
\bigvee_{a \in L} a \supset \bigvee_{r \in R^-(L)} \theta_L(r)
\]
where \(R^-(L)\) is the set of external support rules of \(L\).
Program Completion

- The *completion* of a basic program $P$, denoted by $\text{Comp}(P)$, consists of the following formulas:
  - $\bigwedge_{A \in \text{pos}(r)} \sigma_A \land \bigwedge_{A \in \text{neg}(r)} \neg \sigma_A \supset \text{head}(r)$, for each $r \in P$;
  - $a \supset \bigvee_{r \in P, \text{head}(r) = a} \left( \bigwedge_{A \in \text{pos}(r)} \sigma_A \land \bigwedge_{A \in \text{neg}(r)} \neg \sigma_A \right)$, for each $a \in \mathcal{A}$.

**Theorem 1**

*Let $P$ be a basic program. A set $M \subseteq \mathcal{A}$ is an $r$-answer set of $P$ iff $M$ is a model of $\text{Comp}(P) \cup \{\text{IF}_P(L) \mid L \text{ is a loop of } P\}$.***
Outline

1 Motivations and Preliminaries

2 Our Work

3 Concluding Remark
Function $ML_0(P, X, S)$

$ML := \emptyset$; $G :=$ the $S$ induced subgraph of $G_P$;

For each strongly connected component $L$ of $G$:

if $R^-(L, X) = \emptyset$ then add $L$ to $ML$

else append $ML_0(P, X, L \setminus \{\text{head}(r) \mid r \in R^-(L, X)\})$ to $ML$.

return $ML$.

Let us define

$$loop_0(P, X) = \bigcup_{L \in ML_0(P, X, A)} \lnot L.$$
An iterative procedure

Input: A basic logic program $P$
Output: A set of literals $X$

1. let $X := \emptyset$, compute the completion of $P$ and convert it to set of clauses $C$;
2. compute the loop formulas of the loops that have no external support rules under $X$, then add them to $C$;
3. apply unit propagation to $C$, append the computed consequences to $X$;
4. go back to step 2 until it does not produce any new consequences.
The above procedure computes the well-founded model in [Wang et al. 2012].

The procedure can be improved to compute the (ultimate) well-founded semantics of [Pelov et al. 2007] for aggregate programs with only monotonic aggregate atoms.

If we add loop formulas of loops that have exactly one external support rule in the above procedure, it computes a super set of the well-founded model.

\[ T^P(X) = UP(comp(P) \cup loop_0(P, X) \cup loop_1(P, X) \cup X). \]
1. Motivations and Preliminaries

2. Our Work

3. Concluding Remark
Some Experiments

- We implement the previous procedure for program $P$ that can be accepted by lparse. It first computes the least fixpoint of the $T_P$ operator, denoted $T(P)$, and then adds $\{\leftarrow \text{not } a \mid a \in T(P)\} \cup \{\leftarrow a \mid \neg a \in T(P)\}$ to $P$.
- We run our procedure on the familiar Hamiltonian Circuit (HC) problem encoded with the following cardinality constraints:
  \[\leftarrow 2\{dhc(X, Y) : arc(X, Y)\}, \text{vertex}(Y).\]
  \[\leftarrow 2\{dhc(X, Y) : arc(X, Y)\}, \text{vertex}(X).\]
- The graphs have the structure:
Experiment Results

<table>
<thead>
<tr>
<th>Problem</th>
<th>smodels</th>
<th>smodels&lt;sub&gt;T&lt;/sub&gt;</th>
<th>clasp</th>
<th>clasp&lt;sub&gt;T&lt;/sub&gt;</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>10x10</td>
<td>31.86</td>
<td>17.12</td>
<td>0.05</td>
<td>0.02</td>
<td>0.05</td>
</tr>
<tr>
<td>10x20</td>
<td>-</td>
<td>175.28</td>
<td>0.04</td>
<td>0.06</td>
<td>0.11</td>
</tr>
<tr>
<td>10x30</td>
<td>-</td>
<td>452.39</td>
<td>0.07</td>
<td>0.07</td>
<td>0.18</td>
</tr>
<tr>
<td>10x40</td>
<td>-</td>
<td>-</td>
<td>0.10</td>
<td>0.11</td>
<td>0.26</td>
</tr>
<tr>
<td>10x50</td>
<td>-</td>
<td>-</td>
<td>0.22</td>
<td>0.19</td>
<td>0.31</td>
</tr>
<tr>
<td>20x10</td>
<td>-</td>
<td>-</td>
<td>18.45</td>
<td>5.76</td>
<td>0.44</td>
</tr>
<tr>
<td>20x20</td>
<td>-</td>
<td>-</td>
<td>113.89</td>
<td>91.71</td>
<td>0.93</td>
</tr>
<tr>
<td>20x30</td>
<td>-</td>
<td>-</td>
<td>96.12</td>
<td>18.83</td>
<td>1.34</td>
</tr>
<tr>
<td>20x40</td>
<td>-</td>
<td>-</td>
<td>77.22</td>
<td>2.52</td>
<td>1.89</td>
</tr>
<tr>
<td>20x50</td>
<td>-</td>
<td>-</td>
<td>422.37</td>
<td>201.61</td>
<td>2.38</td>
</tr>
</tbody>
</table>

Table: Run-time Data for smodels and clasp.

- $M \times N$: a graph with $N$ copies of the complete graph with $M$ nodes.
- It shows that for most programs, information from $T(P)$ makes both smodels and clasp run faster, when lookahead operators are turned off.
Our contribution

- We extended the idea to basic logic programs with abstract constraint atoms.
- We gave algorithms for computing loop formulas of loops with at most one external support rule for basic logic programs.
- We considered how they, together with the program completion, can be used to deduce useful consequences of a logic program under unit propagation.
- We related the consequences computed here to the well-founded semantics in [Wang et al. 2012] and [Pelov et al. 2007].
- We showed that the loop formula approach for basic logic programs with c-atoms can also benefit answer set computation for some problems with a certain type of structure.
Thank you!
References

Wang, Y., Lin, F., Zhang, M., and You, J.
A well-founded semantics for basic logic programs with arbitrary abstract constraint atoms.
In Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI-12).

Pelov, N., Denecker, M., and Bruynooghe, M.
Well-founded and stable semantics of logic programs with aggregates.
Theory and Practice of Logic Programming 7, 3, 301–353.