Answer Set Programs with Queries over Subprograms

Christoph Redl
redl@kr.tuwien.ac.at

July 4, 2017

Motivation

Answer Set Programming is a well-known declarative problem solving approach.

Outline

1. Motivation
2. The Saturation Technique and its Restrictions
3. Deciding Inconsistency of Normal Programs in Disjunctive ASP
4. Query Answering over Subprograms
5. Discussion
6. Conclusion

Motivation

Answer Set Programming is a well-known declarative problem solving approach.

An ASP program consists of rules of form

\[ a_1 \lor \cdots \lor a_n \leftarrow b_1, \ldots, b_m, \neg b_{m+1}, \ldots, \neg b_n, \]

An interpretation \( I \) is a set of ground atoms; it is an answer set of a ground program \( P \), if \( I \) is a \( \subseteq \)-minimal model of the reduct \( \Pi = \{ H(r) \leftarrow B^+(r) \mid r \in \Pi, I \not\models b \} \) for all \( b \in B^-(r) \).

Semantics of non-ground programs is defined via a grounding, i.e., replacement of all variables by all constants in all possible ways.

Two (Related) Restrictions

- Meta-reasoning about the answer sets of a (sub)program within another (meta)program not inherently supported.
- Despite \( \Sigma^P_2 \)-completeness of disjunctive ASP solving problems from the first level of the polynomial hierarchy is sometimes tricky.
**Motivation**

**Two (Related) Restrictions**
- Meta-reasoning about the answer sets of a (sub)program within another (meta-)program not inherently supported.
- Despite \( \Sigma^P_2 \)-completeness of disjunctive ASP, solving problems from the first level of the polynomial hierarchy within a program is difficult.

**Contribution**
- An encoding to decide inconsistency of a normal program within a (disjunctive) program.
- An encoding for query answering over a normal program within another program.

---

**The Saturation Technique**

**Basic idea**
- Exploits disjunctions with head-cycles to solve \( \text{coNP} \)-hard problems within ASP.
The Saturation Technique and its Restrictions

The Saturation Technique

Basic idea

Exploits disjunctions with head-cycles to solve coNP-hard problems within ASP.

(Based on the hardness proof of disjunctive ASP [Eiter and Gottlob, 1995].)

Typical use case:

Check if a certain property holds for all objects in a certain domain.

Example

Check if a graph is not 3-colorable.

Consider $P_{\text{non3col}} = F \cup P_{\text{guess}} \cup P_{\text{check}} \cup P_{\text{sat}}$ where

$P_{\text{guess}} = \{ r(X) \lor g(X) \lor b(X) \leftarrow \text{node}(X) \}$

$P_{\text{check}} = \{ \text{sat} \leftarrow c(X), c(Y), \text{edge}(X,Y) \mid c \in \{ r, g, b \} \}$

$P_{\text{sat}} = \{ c(X) \leftarrow \text{node}(X), \text{sat} \mid c \in \{ r, g, b \} \}$

Let $A_0 = P_{\text{non3col}}$.

Is has the answer set $\text{Isat} = A_0$ if the graph is not 3-colorable.

Otherwise its answer sets are proper subsets of $\text{Isat}$ and represent 3-colorings.

Restrictions

Although any problem in coNP can be polynomially reduced to brave reasoning over disjunctive ASP, the reduction is not always obvious.

Example

Check if a graph is not 3-colorable.

Consider $P_{\text{non3col}} = F \cup P_{\text{guess}} \cup P_{\text{check}} \cup P_{\text{sat}}$ where

$P_{\text{guess}} = \{ r(X) \lor g(X) \lor b(X) \leftarrow \text{node}(X) \}$

$P_{\text{check}} = \{ \text{sat} \leftarrow c(X), c(Y), \text{edge}(X,Y) \mid c \in \{ r, g, b \} \}$

$P_{\text{sat}} = \{ c(X) \leftarrow \text{node}(X), \text{sat} \mid c \in \{ r, g, b \} \}$

Let $A_0 = P_{\text{non3col}}$.

Is has the answer set $\text{Isat} = A_0$ if the graph is not 3-colorable.

Otherwise its answer sets are proper subsets of $\text{Isat}$ and represent 3-colorings.
The Saturation Technique

Restrictions

- Although any problem in coNP can be polynomially reduced to brave reasoning over disjunctive ASP, the reduction is not always obvious.
- In particular, the saturation encoding cannot use default-negation.
- ⇒ Checks which involve default-negations must be rewritten.

Example

Check if a graph has no vertex cover \( S \) with size \( |S| \leq k \) for some integer \( k \).

Consider \( P_c \), consisting of facts \( F \) over node and edge and the following parts:

\[
P_{true} = \{ \text{not}(\text{node}(X)) \}
\]
\[
P_{false} = \{ \text{not}(\text{node}(X)) \}
\]
\[
\text{Psat} = \{ \text{not}(\text{sat}(X)), \text{sat}(X) \}
\]

This encoding does not work as desired because model \( D_{sat}(P_{true}) \) is unavailable.

The Saturation Technique

Restrictions

- Although any problem in coNP can be polynomially reduced to brave reasoning over disjunctive ASP, the reduction is not always obvious.
- In particular, the saturation encoding cannot use default-negation.
- ⇒ Checks which involve default-negations must be rewritten.

Example

Check if a graph has no vertex cover \( S \) with size \( |S| \leq k \) for some integer \( k \).

Consider \( P_c \), consisting of facts \( F \) over node and edge and the following parts:

\[
P_{true} = \{ \text{not}(\text{node}(X)) \}
\]
\[
P_{false} = \{ \text{not}(\text{node}(X)) \}
\]
\[
\text{Psat} = \{ \text{not}(\text{sat}(X)), \text{sat}(X) \}
\]

This encoding does not work as desired because model \( D_{sat}(P_{true}) \) is unavailable.

The Saturation Technique

Restrictions

- Although any problem in coNP can be polynomially reduced to brave reasoning over disjunctive ASP, the reduction is not always obvious.
- In particular, the saturation encoding cannot use default-negation.
- ⇒ Checks which involve default-negations must be rewritten.

Example

Check if a ground normal ASP program \( P \) is inconsistent.

Attempt:

\[
P = \{ \text{not}(\text{node}(X)) \} \}
\]
\[
\text{(1)}
\]
\[
\text{(2)}
\]
\[
\text{(3)}
\]
\[
\text{(4)}
\]
\[
\text{(5)}
\]
\[
\text{(6)}
\]
\[
\text{(7)}
\]
The Saturation Technique and its Restrictions

The Saturation Technique

Example

Decide if a ground normal ASP program $P$ is inconsistent.

Attempt:

1. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
2. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
3. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
4. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
5. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
6. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
7. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
8. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
9. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
10. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
11. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
12. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
13. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
14. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
15. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
16. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
17. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
18. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
19. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
20. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
21. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$
22. $\{\text{false} \land \text{false} \land \text{false} \land \text{false} \}$

However, the comparison of the least model of the reduct to the original guess in rule uses default negation.

Outline

Motivation

The Saturation Technique and its Restrictions

Deciding Inconsistency of Normal Programs in Disjunctive ASP

Query Answering over Subprograms

Discussion

Conclusion

A Meta-Program for Propositional Programs

Basic idea

- Use a meta-program $M$ to simulate an ASP solver in disjunctive ASP.
- Encode the ASP program $P$ to check for inconsistency as facts $M_P$.
- Evaluate $M$ over $M_P$ to identify inconsistency or the answer sets of $P$.
- (Alternative encodings exist, see e.g. [Elbers and Pollehn, 2006].)

Proposition

For any ground normal logic program $P$, we have that:
1. If $P$ is inconsistent, $M$ over $M_P$ has exactly one answer set which contains $\text{false}$. and
2. If $P$ is consistent, $M$ over $M_P$ has at least one answer set and none of the answer sets of $M$ contain $\text{false}$.}

A Meta-Program for Propositional Programs

Basic idea

- Use a meta-program $M$ to simulate an ASP solver in disjunctive ASP.
- Encode the ASP program $P$ to check for inconsistency as facts $M_P$.
- Evaluate $M$ over $M_P$ to identify inconsistency or the answer sets of $P$.
- (Alternative encodings exist, see e.g. [Elbers and Pollehn, 2006].)

A Meta-Program for Propositional Programs

Definition

We define the meta-program $M = M_{\text{AS}} \cup M_{\text{m}} \cup M_{\text{head}} \cup M_{\text{inReduct}}$, where:

1. $M_{\text{AS}} = \{\text{true}\}$
2. $M_{\text{m}} = \{\text{false}\}$
3. $M_{\text{head}} = \{\text{false}\}$
4. $M_{\text{inReduct}} = \{\text{false}\}$

A Meta-Program for Propositional Programs

Evaluation

We evaluate $M$ over $M_P$ to check for inconsistency:

1. If $M$ contains $\text{false}$, then $P$ is inconsistent.
2. If $M$ contains $\text{true}$, then $P$ is consistent.

A Meta-Program for Propositional Programs

Implementation

We implement $M$ using an ASP solver in disjunctive ASP:

1. Use a meta-program $M$ to simulate an ASP solver in disjunctive ASP.
2. Encode the ASP program $P$ to check for inconsistency as facts $M_P$.
3. Evaluate $M$ over $M_P$ to identify inconsistency or the answer sets of $P$.
4. (Alternative encodings exist, see e.g. [Elbers and Pollehn, 2006].)
Deciding Inconsistency of Normal Programs in Disjunctive ASP

A Meta-Program for Propositional Programs

To lift the idea to non-ground programs, we exploit function symbols: predicates in the input program become function symbols in the meta-program.

Definition
For a ground normal logic program \( P \) we let:
\[
\begin{align*}
\mathcal{M}_e &= \{ (\text{not } p(X), c) : (\not c p(X)) \in P \} \\
\cup \{ (\text{not } f(X), c) : (\not c f(X)) \in P \}
\end{align*}
\]

A Meta-Program for Non-Ground Programs

To lift the idea to non-ground programs, we exploit function symbols: predicates in the input program become function symbols in the meta-program.

Definition
For a (ground or non-ground) normal logic program \( P \) we let:
\[
\begin{align*}
\mathcal{M}_e &= \{ (\text{not } p(X), c) : (\not c p(X)) \in P \} \\
\cup \{ (\text{not } f(X), c) : (\not c f(X)) \in P \}
\end{align*}
\]
Query Answering over Subprograms

We reduce brave and cautious queries over subprograms to inconsistency checking.

Observation:

Definition

A ground query atom is of form \( S \sqcap_t q \), where \( t \in \{ b, c \} \) determines the type of the query, \( S \) is a normal logic (sub)program, and \( q \) is a query over \( S \).

Query atoms may occur in bodies of ASP programs in place of ordinary atoms.

Definition

A ground query atom is of form \( S \sqcap r \), where \( r \in \{ \langle a, v \rangle \} \) determines the type of the query, \( S \) is a normal logic (sub)program, and \( q \) is a query over \( S \).

Query atoms may occur in bodies of ASP programs in place of ordinary atoms.

Initiation:

\[ S \sqcap r, \ \text{resp.} \ S \sqcap \neg r, \ \text{is true (wrt. all interpretations \( I \) of \( P \), \( I \subseteq \text{AS}(P) \sqcap I \)).} \]
Query Answering: Language Extension

Now we extend ASP with queries over subprograms:

Definition

A ground query atom is of form $S \leftarrow t q$, where $t \in \{b, c\}$ determines the type of the query, $S$ is a normal logic (sub)program, and $q$ is a query over $S$.

Query atoms may occur in bodies of ASP programs in place of ordinary atoms.

Intuition:

$S \leftarrow b q$ resp. $S \leftarrow c q$ is true (wrt. all interpretations $I$ of $P$) if $S \models b q$ resp. $S \models c q$.

Formally the semantics of such a program $P$ uses the following translation:

$$\langle P \rangle = P \cup \{ \text{¬} \bar{l} \mid l \in q \}$$

For a logic program $P$ with query atoms, the answer sets of $P$ and $\langle P \rangle$, projected to the atoms in $P$, coincide.

Example

Suppose $P_{\text{guess}}$ guesses all edge selections in a graph and $P_{\text{check}}$ derives invalid if the current selection is not a Hamiltonian cycle.

Then

$$P = \{ \text{noHamiltonian} \leftarrow P_{\text{guess}} \cup P_{\text{check}} \}$$

(and thus $\langle P \rangle$) has an answer set containing noHamiltonian if and only if the graph at hand does not contain a Hamiltonian cycle.

Otherwise the program has at least one answer set but none of the answer sets contains atom noHamiltonian.

Query Answering: Checking Conditions with Default-Negation

In general:

Steps

1. Let program $P_{\text{guess}}$ span a search space of all objects to check.
2. Let $P_{\text{check}}$ check if the current guess satisfies the criteria and derive atom ok in this case.
3. Instead of manually saturating whenever ok is true, just check if it is cautiously entailed by $P_{\text{guess}} \cup P_{\text{check}}$.

Outline

- Motivation
- The Saturation Technique and its Restrictions
- Deciding Inconsistency of Normal Programs in Disjunctive ASP
- Query Answering over Subprograms
- Discussion
- Conclusion

Discussion

Alternative Approaches

- Nested HEX programs [Eiter et al., 2013]:
  Based on the ASP-extension of HEX-programs (with access to external sources) rather than plain ASP.
Discussion

Alternative Approaches

- Nested HEX-programs [Eiter et al., 2013]: Based on the ASP-extension of HEX programs (with access to external sources) rather than plain ASP.
- Modular ASP [Tar et al., 2005]: Programs consisting of components but no compilation into a single program.


Modular ASP [Tari et al., 2005]: Programs consisting of components but no compilation into a single program.


Manifold programs [Faber and Woltran, 2011]: Compilation-based using weak constraints.


Stable-unstable semantics [Bogaerts et al., 2016]: Similar goal (isolating the oracle program) and generalize to the PH, but no dedicated query atoms.


Encoding is also related to debugging approaches (e.g. [Gebser et al., 2008, Oetsch et al., 2010]): Rather than explaining inconsistency we exploit it for query answering.
Conclusion

Two Restriction of Answer Set Programming

- No meta-reasoning over collections of answer sets.
- Limitations of the saturation technique: difficult to use for ASP laymen and restricted use of default-negation.

Contribution and Solution

- Encoding for deciding inconsistency of a normal program.
- Encoding for query answering over a normal program.
- A language extension of ASP program with dedicated query atoms.
- More user-friendly alternative to saturation.

Future Work

- Extension to non-ground queries.
- Implementation and application for nested HEX-program evaluation.

References I


References II


References III