Verifying the Equivalence of Disjunctive Logic Programs

Emilia Oikarinen
Helsinki University of Technology
Laboratory for Theoretical Computer Science
emilia.oikarinen@hut.fi

Based on Emilia Oikarinen and Tomi Janhunen, Verifying the Equivalence of Logic Programs in the Disjunctive Case. In the Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning.

Motivation

• In answer set programming (ASP) a problem at hand is solved by constructing a logic program whose answer sets correspond to the solutions of the problem.
• There can be several versions of the logic program formalizing the problem.
• A meta-level problem: how to ensure that different encodings yield the same output i.e. have the same answer sets?

Outline

• Motivation: Equivalence of Logic Programs
• Disjunctive Logic Programs: Syntax and Semantics
• Translation-based Verification Method
• Experiments
• Conclusions

Motivation Cont’d

• We consider the following notion of equivalence
  • Logic programs $P$ and $Q$ are equivalent ($P \equiv Q$) $\iff$ $P$ and $Q$ have exactly the same answer sets.

• We consider equivalence of disjunctive logic programs.

• This is generalization of previous work where we developed an automated translation-based method for verifying the equivalence of programs supported by the SMODELS system.

• Note that deciding $P \equiv Q$ for finite propositional disjunctive programs is $\Pi^P_2$-hard.
Disjunctive Logic Programs

- A (propositional) disjunctive logic program (DLP) $P$ is a set of rules of the form
  \[ a_1 | \ldots | a_n \leftarrow b_1, \ldots, b_m, \lnot c_1, \ldots, \lnot c_k, \]
  where $a_1, \ldots, a_n, b_1, \ldots, b_m, c_1, \ldots, c_k$ are propositional atoms and $n, k, m$ are natural numbers.
- A shorthand: $A \leftarrow B, \lnot C$.
- Program $P$ is normal if $n = 1$ for each rule of $P$.
- Program $P$ is positive if $k = 0$ for each rule of $P$.

Satisfaction Relation and Minimal Models

- The Herbrand base $\text{Hb}(P)$ is the set of atoms appearing in $P$.
- An interpretation $I \subseteq \text{Hb}(P)$ of $P$ defines which atoms $a \in \text{Hb}(P)$ are true ($a \in I$) and which are false ($a \notin I$).
- An interpretation $I$ is a (classical) model of $P$ ($I \models P$) if for each $A \leftarrow B, \lnot C \in P$, $B \subseteq I$ and $C \cap I = \emptyset$ imply $A \cap I \neq \emptyset$.
- $M$ is a minimal model of $P$, if there is no $M' \subset M$ such that $M' \models P$. The set of minimal models of $P$ is denoted by $\text{MM}(P)$.

Stable Model Semantics

- Given a DLP $P$ and $M \subseteq \text{Hb}(P)$, the Gelfond-Lifschitz reduct of $P$ is a positive program
  \[ P_M = \{ A \leftarrow B \mid A \leftarrow B, \lnot C \in P \text{ and } M \cap C = \emptyset \}. \]
- $M$ is a stable model of $P$ if $M \in \text{MM}(P_M)$.
- We denote the set of stable models of $P$ by $\text{SM}(P)$.

Example. Consider $P = \{ a \mid b \leftarrow \lnot b, b \leftarrow \lnot a \}$ and $M = \{ a \}$. Now, $P_M = \{ a \mid b \leftarrow \}$ and $\text{MM}(P_M) = \{ \{ a \}, \{ b \} \}$. Thus $M \in \text{SM}(P)$.

Verifying Equivalence

- We assume that $\text{Hb}(P) = \text{Hb}(Q)$ (without loss of generality).
- We consider a translation $\text{TR}(P, Q)$ such that $\text{TR}(P, Q)$ has a stable model $\iff \exists M \in \text{SM}(P) \text{ s.t. } M \notin \text{SM}(Q)$. Thus, $P \equiv Q \iff \text{SM}(\text{TR}(P, Q)) = \emptyset \text{ and } \text{SM}(\text{TR}(Q, P)) = \emptyset$.
- We can distinguish two types of counter-examples for equivalence.
  - T1: $\langle M, M \rangle$ s.t. $M \in \text{SM}(P)$ and $M \notin Q_M$.
  - T2: $\langle M, M' \rangle$ s.t. $M \in \text{SM}(P)$, $M \models Q_M$, $M' \subset M$ and $M' \models Q_M$. 
## Two-Phased Translation

- Since there are two types of counter-examples for equivalence, testing can be performed in two phases.
  - **Phase 1:** \( \text{SM}(\text{TR}_1(P, Q)) \neq \emptyset \) \iff \exists M \in \text{SM}(P) \text{ s.t. } M \not\models Q_M, \)
i.e. there exists a counter-example of type T1.
  - **Phase 2 (if \( \text{SM}(\text{TR}_1(P, Q)) = \emptyset \)):** \( \text{SM}(\text{TR}_2(P, Q)) \neq \emptyset \) \iff \exists M \in \text{SM}(P) \text{ s.t. } M \not\in \text{MM}(Q_M), \)
i.e. there exists a counter-example of type T2.
- \( \text{TR}_1(P, Q) \) and \( \text{TR}_2(P, Q) \) can easily be obtained from \( \text{TR}(P, Q) \).

## Experiments

- The translation functions have been implemented in C under Linux and a *naive cross-checking approach* as a shell script.
- The current implementation \( \text{DLPEQ} \) is available in the web:
  
  http://www.tcs.hut.fi/Software/lpeq/
- The performance of the naive and the two translation-based approaches was compared in several experiments.
- A two-way search of counter-examples was performed in any case.
- \( \text{GN} \) was used for the computation of stable models.

## Disjunctive Random 3-SAT

- Finding a minimal model of a random 3-SAT instance containing specified atoms as a test problem.
- Encoding as DLPs that solve an instance of a random 3-SAT problem and additional rules for random atoms \( c_i \), for \( i = 1, \ldots, [2v/100] \), where \( v \) is the number of atoms.
- A fixed clauses to variables ratio \( c/v = 3.5 \).
- We test the equivalence of each program \( P \) against a variant \( P' \) obtained by dropping a random rule from \( P \).

### Results: Disjunctive Random 3-SAT

- **Time (s)** vs. number of variables
- **Number of variables**
- **Time (s)**
- **NAIVE**
- **DLPEQ**
- **DLPEQ2**
Conclusions

- Two translation-based methods and an implementation for verifying the equivalence of DLPs have been presented.
- In many cases, the time needed for computations is less than in a naive approach of cross-checking the stable models.
- If programs have no/few stable models, then the naive approach can become superior to the translation-based ones.
- Two-phased translation is faster than the one-phased one.
- Future work: experiments using real-life problems, extension to other classes of logic programs, other notions of equivalence.