ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-1139

From Disjunctive to Normal Logic Programs via Unfolding and Shifting

Yi Zhou [1]

Abstract. We show that every propositional disjunctive logic program under the answer set semantics can be equivalently transformed into a normal one via unfolding and shifting. More precisely, after iteratively applying the unfolding operator for some rules in a disjunctive program, its shifted program, which is a normal program, must have the same answer sets as the original disjunctive program.

1 Introduction

In Answer Set Programming (ASP), a fundamental issue arises whether disjunctive logic programs are more expressive than normal ones. The answer is "yes" from a computational complexity point of view. However, from an answer set point of view, the answer is "no". In fact, Eiter et al. [4] showed that, for any given propositional disjunctive program, there always exists an equivalent normal program under the answer set semantics, i.e., they have the same answer sets. Nevertheless, Eiter et al.'s construction is essentially semantic in the sense that they first compute all the answer sets of the original disjunctive program, from which they reconstruct an equivalent normal program.

In this paper, we propose a syntactic transformation from DLP to NLP. Interestingly, our transformation is based on two classical operators in the literature, namely unfolding [3, 8] and shifting [6]. We prove that, after iteratively applying the unfolding method for some rules in a disjunctive program, its shifted program, which is a normal program, must have the same answer sets as the original disjunctive program.

2 Preliminaries

A *disjunctive logic program* is a set of *rules* of the following form:

$$p_1; \ldots; p_m \leftarrow p_{m+1}, \ldots, p_n, \texttt{Not}\ p_{n+1}, \ldots, \texttt{Not}\ p_l, \qquad (1)$$

where $0 \leq m \leq n \leq l$, and every $p_i, (1 \leq i \leq l)$ is an atom or $\bot$. The head, body, positive and negative bodies are defined as usual. A rule of form (1) is said to be *normal* if $m = 1$.

Let $X \subseteq Atom$ be a set of atoms and Π be a program. We say that X is an *answer set* of Π if X is a minimal model of Π^X, where Π^X is the reduct of Π relative to X. For convenience, we use $AS(\Pi)$ to denote the collection of all answer sets of Π. We say that two programs are *equivalent* if they have the same set of answer sets.

Let r_1 and r_2 be the following two rules

$$\begin{aligned} r_1 &= a; H_1 \leftarrow B_1 \\ r_2 &= H_2 \leftarrow a, B_2, \end{aligned}$$

where a is an atom such that $a \in Head(r_1) \cap Pos(r_2)$. The rule obtained by *unfolding* a on r_1 and r_2, denoted by $\texttt{Unfold}(r_1, r_2, a)$, is the following rule

$$H_1; H_2 \leftarrow B_1, B_2.$$

Let Π be a program and r a rule in it. The result of *unfolding* r in Π, denoted by $\texttt{Unfold}(\Pi, r)$, is the following program

$$\Pi \backslash \{r\} \cup \{\texttt{Unfold}(r_1, r, a) \mid r_1 \in \Pi, a \in Pos(r) \cap Head(r_1)\}.$$

A weaker version is to retain the rule r in the new program. We use $\texttt{Unfold}^W(\Pi, r)$ to denote $\texttt{Unfold}(\Pi, r) \cup \{r\}$.

Theorem 1 *[3] Let Π be a program and r a rule in it. Then,*

$$AS(\Pi) = AS(\texttt{Unfold}(\Pi, r)) = AS(\texttt{Unfold}^W(\Pi, r)).$$

Let r be a rule of form (1). The result of *shifting* r, denoted by $\texttt{Shift}(r)$, is the set of following rules [6]:

$$\begin{aligned} p_1 \leftarrow\ & p_{m+1}, \ldots, p_n, \texttt{Not}\ p_{n+1}, \ldots, \texttt{Not}\ p_l, \\ & \qquad\qquad \texttt{Not}\ p_1, \texttt{Not}\ p_3, \ldots, \texttt{Not}\ p_m, \\ & \ldots \quad \ldots \quad \ldots \quad \ldots \\ p_m \leftarrow\ & p_{m+1}, \ldots, p_n, \texttt{Not}\ p_{n+1}, \ldots, \texttt{Not}\ p_l, \\ & \qquad\qquad \texttt{Not}\ p_1, \ldots, \texttt{Not}\ p_{m-1}. \end{aligned}$$

Let Π be a program. The result of *shifting* Π, denoted by $\texttt{Shift}(\Pi)$, is $\bigcup_{r \in \Pi} \texttt{Shift}(r)$. Clearly, for any disjunctive program Π, $\texttt{Shift}(\Pi)$ is a normal program.

Theorem 2 *[6] Let Π be a program. Then,*

$$AS(\texttt{Shift}(\Pi)) \subseteq AS(\Pi).$$

However, the converse of Theorem 2 does not hold in general.

3 From DLP to NLP via Unfolding and Shifting

In this section, we show that any finite propositional disjunctive program can be equivalently transformed into a normal program via unfolding and shifting.

We first show that for programs closed under unfolding, they have the same set of answer sets as their shifted program. Let Π be a program. We say that Π is *closed under unfolding* if for any $r_1, r_2 \in \Pi$, $a \in Atom$ such that $a \in Head(r_1) \cap Pos(r_2)$, $\texttt{Unfold}(r_1, r_2, a) \in \Pi$.

[1] Artificial Intelligence Research Group, School of Computing, Engineering and Mathematics, University of Western Sydney, Australia.

Theorem 3 *If a disjunctive program* Π *is closed under unfolding, then*

$$AS(\texttt{Shift}(\Pi)) = AS(\Pi).$$

Now we consider how to translate a disjunctive program to a normal one via unfolding and shifting. Let Π be a program. An *unfolding sequence* of Π, is a sequence of programs $\Pi_0, \Pi_1, \ldots, \Pi_k, \ldots$ such that $\Pi_0 = \Pi$ and for all i, Π_{i+1} is

$$\Pi_i \cup \{\texttt{Unfold}(r_1, r_2, a) \mid r_1, r_2 \in \Pi_i, a \in Head(r_1) \cap Pos(r_2)\}.$$

Theorem 4 *Let* $\Pi_0, \Pi_1, \ldots, \Pi_k, \ldots$ *be an unfolding sequence of a program* Π*. For any* $i < j$*,*

$$AS(\Pi_i) = AS(\Pi_j).$$

Next, we consider to shift the intermediate programs Π_i in an unfolding sequence.

Theorem 5 *Let* $\Pi_0, \Pi_1, \ldots, \Pi_k, \ldots$ *be an unfolding sequence of a program* Π*. For any* $i < j$*,*

$$AS(\texttt{Shift}(\Pi_i)) \subseteq AS(\texttt{Shift}(\Pi_j)).$$

Finally, we prove our main theorem any unfolding sequence of a disjunctive program will eventually lead to an equivalent normal program.

Theorem 6 *For any unfolding sequence* $\Pi_0, \Pi_1, \ldots, \Pi_k, \ldots$ *of a disjunctive program* Π*, there exists* k *such that*

$$AS(\texttt{Shift}(\Pi_k)) = AS(\Pi).$$

Proof Since the unfolding sequence is monotonic, both in terms of rules and in terms of answer sets (see Theorem 5), there exists k such that Π_k is closed under unfolding (the number of atoms is finite). Then, by Theorem 3, $AS(\texttt{Shift}(\Pi_k)) = AS(\Pi)$.

In fact, using the ideas of unfolding sequence based on Theorems 5 and 6, one can develop a new way of finding an answer set for disjunctive programs by using an NLP solver. The solver will first call the NLP solver to compute the answer sets of the shifted program of the original disjunctive program. If successful, then by Theorem 5, this answer set must be an answer set of the disjunctive program as well. If failed, then the solver will start the unfolding sequence, i.e., the solver will add to the original program some unfolded rules, and then call the NLP solver again to compute the answer sets of its shifted program. Again, by Theorem 5, any answer set of the shifted program in the unfolding sequence should be an answer set of the original disjunctive program. Finally, guaranteed by Theorem 6, this process will eventually ends.

4 Related Work and Discussions

Both the unfolding operator and the shifting operator are extensively studied in the literature [1, 2, 3, 5, 6, 8]. However, to the best of our knowledge, the combination of these two operators seems to be a new idea. As we have shown in this paper, shifting together with unfolding is powerful enough to translate any disjunctive logic programs into normal ones.

The relationships between disjunctive logic programs and normal programs have also been extensively studied. From a complexity point of view, unless the polynomial hierarchy collapses, there is no polynomial, sound and complete translation from DLP to NLP. However, if we release the condition of polynomial, there are sound and complete translations. For instance, Eiter et al. [4] proposed some semantical characterization of the answer set semantics and showed that every disjunctive logic program under the answer set semantics can be equivalently transformed into a normal one (see Theorem 5 in [4]). However, Eiter et al.'s translation needs to compute all answer sets of the disjunctive program first, from which one can reconstruct an equivalent normal program. In this sense, this translation is exponential and semantic. In this paper, we show that any disjunctive program can be translated into a normal one via unfolding and shifting. This translation can be regarded as a syntactic translation as both the unfolding and the shifting operators are syntactic transformations. Although it still needs to introduce an exponential number of unfolded rules, we can control them cautiously by unfolding sequences. As far as we know, our translation is the first syntactic translation from DLP to NLP.

Janhunen et al. [7] also considered an approach to compute answer sets of disjunctive programs by calling normal ASP solvers. Interesting, this approach is called "unfolding". In their approach, for each disjunctive program Π, a generating (normal) program $Gen(\Pi)$ is constructed to produce candidate models M, then a testing (normal) program $Test(\Pi, M)$ is used to check whether M is indeed an answer set of Π. Nevertheless, this approach is essentially different from ours. First, the term "unfolding" in Janhunen et al.'s approach is simply a word, meaning that NLP solvers are used to solve DLP, which is irrelevant to the unfolding operator. Second, in our approach, there is no testing program as the translation is sound. Third, the generating program $Gen(\Pi)$ is not sound but complete (in some sense), while in our approach, the shifted program of any intermediate program is sound but not necessarily complete.

5 Conclusion

Our main result shows that any propositional disjunctive program under the answer set semantics can be equivalently transformed into a normal one, via two syntactic operators, namely unfolding and shifting. This result is a little surprising as both the unfolding and the shifting operators are extensively studied in the literature. However, the combination of both seems new yet powerful.

REFERENCES

[1] Vernon Asuncion, Fangzhen Lin, Yan Zhang, and Yi Zhou, 'Ordered completion for first-order logic programs on finite structures', *Artif. Intell.*, **177-179**, 1–24, (2012).
[2] Rachel Ben-Eliyahu and Rina Dechter, 'Propositional semantics for disjunctive logic programs', *Ann. Math. Artif. Intell.*, **12**(1-2), 53–87, (1994).
[3] Stefan Brass and Jürgen Dix, 'Semantics of (disjunctive) logic programs based on partial evaluation', *J. Log. Program.*, **40**(1), 1–46, (1999).
[4] Thomas Eiter, Michael Fink, Hans Tompits, and Stefan Woltran, 'On eliminating disjunctions in stable logic programming', in *KR*, pp. 447–458, (2004).
[5] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub, 'Advanced conflict-driven disjunctive answer set solving', in *IJCAI*, (2013).
[6] Michael Gelfond, Halina Przymusinska, Vladimir Lifschitz, and Miroslaw Truszczynski, 'Disjunctive defaults', in *KR*, pp. 230–237, (1991).
[7] Tomi Janhunen, Ilkka Niemelä, Dietmar Seipel, Patrik Simons, and Jia-Huai You, 'Unfolding partiality and disjunctions in stable model semantics', *ACM Trans. Comput. Log.*, **7**(1), 1–37, (2006).
[8] Chiaki Sakama and Hirohisa Seki, 'Partial deduction in disjunctive logic programming', *J. Log. Program.*, **32**(3), 229–245, (1997).