Translating Parallel Circumscription into Disjunctive Logic Programming*

Emilia Oikarinen

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory for Theoretical Computer Science
P.O.Box 5400, FI-02015 TKK, Finland
Emilia.Oikarinen@tkk.fi

Abstract. The semantics of disjunctive logic programs is based on minimal models. This makes atoms appearing in a disjunctive program false by default. In many cases this is highly desirable, but certain problems become awkward to formalize if all atoms are subject to minimization. Lifschitz's parallel circumscription enables the use of varying and fixed atoms which eases the task of knowledge presentation in certain cases. In this paper we show how parallel circumscription can be embedded into disjunctive logic programming using a linear translation.

1 Introduction

A rule-based language allowing disjunctions in the heads of rules is used for knowledge representation in *disjunctive logic programming*. The semantics of disjunctive logic programs is determined by *stable models* [4, 12]. Stable models are minimal with respect to subset inclusion. Thus every atom appearing in a disjunctive logic program is false by default. Often this is highly desirable, but some problems are difficult to formalize if all atoms are minimized.

Parallel circumscription [10] is based on a refined notion of minimality and enables the use of *varying* and *fixed* atoms in addition to the ones that are being minimized. This eases the task of knowledge presentation in certain cases. For example, it is very straightforward to formalize Reiter-style minimal diagnoses [13] for digital circuits using parallel circumscription.

There have been several attempts to embed parallel circumscription into disjunctive logic programming. Although fixed atoms are easy in this respect [2, 5], varying atoms are not fully covered. Earlier approaches either deal with syntactic subclasses of logic programs [3] or have exponential worst-case space complexities [8, 14]. Cadoli et al. [1] achieve linear complexity, but their transformation is about reducing circumscriptive inference to inference from minimal models. Wakaki and Inoue [15] propose an approach for computing *prioritized*

* The research reported is funded by Helsinki Graduate School in Computer Science, Academy of Finland (project #211025) and by grants from Nokia Foundation and Finnish Cultural Foundation.

circumscription (a generalization of parallel circumscription) using a two-phased *generate and test* method.

In this paper we discuss the approach presented in [7]. Contrarily to earlier approaches the translation described is *linear* but *non-modular* and *enables the use of existing implementations* of disjunctive logic programming such as DLV [9] and GnT [6] for the actual search of minimal models.

2 Disjunctive Logic Programs

A (propositional) *disjunctive logic program* (DLP) Π is a set of *rules* of the form

$$a_1 \vee \cdots \vee a_n \leftarrow b_1, \ldots, b_m, {\sim}c_1, \ldots, {\sim}c_k, \tag{1}$$

where $n, m, k \geq 0$ and $a_1, \ldots, a_n$, $b_1, \ldots, b_m$, and $c_1, \ldots, c_k$ are propositional atoms. The *head* of the rule $a_1 \vee \cdots \vee a_n$ is interpreted disjunctively while the rest – forming the *body* of the rule – is interpreted conjunctively. The symbol "$\sim$" denotes *negation as failure*. Intuitively, a rule of the form (1) acts as an inference rule; any of the head atoms $a_1, \ldots, a_n$ can be inferred given that the positive body atoms $b_1, \ldots, b_m$ can be inferred and the negative body atoms $c_1, \ldots, c_k$ cannot. We define *literals* in the standard way using $\sim$ as the connective for negation. For any set of atoms A, we define a set of negative literals ${\sim}A = \{{\sim}a \mid a \in A\}$. Since the order of atoms is insignificant in a rule (1), we use a shorthand $A \leftarrow B, {\sim}C$ where A, B and C are the sets of atoms involved in (1). We drop the symbol "$\leftarrow$" in case of an empty body. An empty head ($n = 0$) is denoted by "$\bot$" and a rule with an empty head is called an *integrity constraint*. A DLP Π is *positive* (PDLP) if and only if $k = 0$ holds for every rule (1) of Π.

2.1 Stable Model Semantics

The *Herbrand base* $\mathrm{Hb}(\Pi)$ of a DLP Π is the set of atoms appearing in Π. An *interpretation* $I \subseteq \mathrm{Hb}(\Pi)$ of Π determines which atoms $a \in \mathrm{Hb}(\Pi)$ are *true* ($a \in I$) and which are *false* ($a \notin I$). An interpretation I is a (classical) *model* of Π, denoted by $I \models \Pi$ if and only if for every rule $A \leftarrow B, {\sim}C$ of Π, $B \subseteq I$ and $C \cap I = \emptyset$ imply $A \cap I \neq \emptyset$, i.e. the satisfaction of the rule body implies that one of the head atoms must also be true.

An interpretation $M \subseteq \mathrm{Hb}(\Pi)$ is a *minimal model* of Π if and only if $M \models \Pi$ and there is no $N \subset M$ such that $N \models \Pi$. We denote the set of minimal models of Π by $\mathrm{MM}(\Pi)$. If Π is a PDLP, then $\mathrm{MM}(\Pi)$ determines the standard minimal model semantics of Π. The semantics of DLPs involving negation as failure is defined through *stable models* [4, 12].

Definition 1. *Given a DLP Π and an interpretation $M \subseteq \mathrm{Hb}(\Pi)$, the Gelfond-Lifschitz reduct of Π is a PDLP*

$$\Pi^M = \{A \leftarrow B \mid A \leftarrow B, {\sim}C \in \Pi \text{ and } M \cap C = \emptyset\}. \tag{2}$$

An interpretation $M \subseteq \mathrm{Hb}(\Pi)$ is a stable model of Π if and only if $M \in \mathrm{MM}(\Pi^M)$. The set of stable models of Π is denoted by $\mathrm{SM}(\Pi)$.

3 Parallel Circumscription

We formulate parallel circumscription in the propositional case and assume that the syntax of PDLPs is used instead of arbitrary propositional sentences. Parallel circumscription is based on a refined notion of minimality partitioning atoms in three disjoint categories.

Definition 2. *Let Π be a PDLP and let $V, F \subseteq \mathrm{Hb}(\Pi)$ be the sets of varying and fixed atoms, respectively, and $V \cap F = \emptyset$. A model $M \models \Pi$ is $\langle V, F\rangle$-minimal if and only if there is no $N \models \Pi$ such that (i) $N \setminus (V \cup F) \subset M \setminus (V \cup F)$ and (ii) $N \cap F = M \cap F$.*

The idea is that the atoms in $\mathrm{Hb}(\Pi) \setminus (V \cup F)$ are minimized, while the truth values of the atoms in V may *vary* freely and the truth values of the atoms in F are kept *fixed*. Notice that the $\langle \emptyset, \emptyset\rangle$-minimal models of a PDLP are its minimal models.

4 Translating Parallel Circumscription into a DLP

In this section we consider the problem of translating parallel circumscription into disjunctive logic programming. We proceed by introducing the subtranslations $\mathrm{Tr_F}(\cdot)$ and $\mathrm{Tr_V}(\cdot)$ used to remove fixed and varying atoms, respectively. The translations are then combined to obtain a bijective correspondence between the $\langle V, F\rangle$-minimal models of a PDLP and the stable models of its translation.

We use a technique proposed by De Kleer and Konolige [2] to remove the fixed atoms. Let Π be a PDLP and $V, F \subseteq \mathrm{Hb}(\Pi)$. A new atom f' is introduced for each $f \in F$. The translation for removing fixed atoms is

$$\mathrm{Tr_F}(\Pi) = \Pi \cup \{f \vee f'.\ \ \bot \leftarrow f, f'.\ \ \mid f \in F\}$$

with $(\mathrm{Hb}(\Pi) \setminus V) \cup \{f' \mid f \in F\}$ as the set of minimized atoms. There is a bijective relationship between the $\langle V, F\rangle$-minimal models of Π and the $\langle V, \emptyset\rangle$-minimal models of $\mathrm{Tr_F}(\Pi)$. Furthermore, $M' = M \cup \{f' \mid f \in F \text{ and } f \notin M\}$ is a $\langle V, \emptyset\rangle$-minimal model of $\mathrm{Tr_F}(\Pi)$ iff M is a $\langle V, F\rangle$-minimal model of Π.

The translation $\mathrm{Tr_V}(\cdot)$ for removing varying atoms is more involved. We use a slightly optimized version of the translation proposed in [7]. It is worth noticing that while the translation is linear with respect to the length of the original program, it is non-modular. We assume that fixed atoms have already been removed using the translation $\mathrm{Tr_F}(\cdot)$. The goal is to obtain a bijective relationship between the $\langle V, \emptyset\rangle$-minimal models of PDLP Π and the stable models of its translation $\mathrm{Tr_V}(\Pi)$.

The translation function $\mathrm{Tr_V}(\cdot)$ introduces new atoms not appearing in $\mathrm{Hb}(\Pi)$ as follows. For each $a \in \mathrm{Hb}(\Pi)$, the complement $\overline{a}$ of a expresses the falsity of a. Another renamed copy a^* of each $a \in \mathrm{Hb}(\Pi)$ is needed in formulating a test for $\langle V, \emptyset\rangle$-minimality along with a vector of new atoms $d_1, \ldots, d_n$ for the minimized atoms $P = \mathrm{Hb}(\Pi) \setminus V = \{a_1, \ldots, a_n\}$. A new atom u for unsatisfiability is

needed for the unsatisfiability check used to encode the test for $\langle V, \emptyset \rangle$-minimality. Given a set of atoms $A \subseteq \mathrm{Hb}(\Pi)$, we introduce shorthands $\overline{A}$ and A^* for the sets $\{\overline{a} \mid a \in A\}$ and $\{a^* \mid a \in A\}$, respectively.

Definition 3. *Let Π be a PDLP, $V \subseteq \mathrm{Hb}(\Pi)$ a set of varying atoms, and define $P = \mathrm{Hb}(\Pi) \setminus V = \{a_1, \ldots, a_n\}$. The translation $\mathrm{Tr_V}(\Pi)$ contains the following rules:*

1. $a \leftarrow \sim\overline{a}$ *for each* $a \in V$ *and* $\overline{a} \leftarrow \sim a$ *for each* $a \in \mathrm{Hb}(\Pi)$;
2. $(A \setminus V) \leftarrow (B \setminus V), \sim(A \cap V), \sim\overline{B \cap V}$ *for each rule* $A \leftarrow B$ *in* Π;
3. $A^* \cup \{u\} \leftarrow B^*$ *for each rule* $A \leftarrow B$ *in* Π;
4. $d_1 \vee \cdots \vee d_n \vee u$;
5. $u \leftarrow d_i, \sim a_i$ *and* $u \leftarrow a_i^*, \sim a_i$ *for each* $1 \leq i \leq n$;
6. $u \leftarrow d_i, a_i^*, \sim\overline{a_i}$ *and* $u \vee d_i \vee a_i^* \leftarrow \sim\overline{a_i}$ *for each* $1 \leq i \leq n$;
7. $a^* \leftarrow u$ *for each* $a \in \mathrm{Hb}(\Pi)$;
8. $d_i \leftarrow u$ *for each* $1 \leq i \leq n$; *and*
9. $\bot \leftarrow \sim u$.

The translation works as follows. While taking into account that atoms in P are minimized, the rules in items (1.) and (2.) choose an arbitrary model candidate M such that $M \models \Pi$. The rules in item (3.) create a renamed copy of Π for checking the $\langle V, \emptyset \rangle$-minimality of M. The rules in items (4.)–(6.) enforce that any potential counter-model N for the $\langle V, \emptyset \rangle$-minimality of M (expressed in $\mathrm{Hb}(\Pi)^*$ rather than in $\mathrm{Hb}(\Pi)$) satisfies $N \cap P \subset M \cap P$. The atoms d_i represent the difference between $M \cap P$ and $N \cap P$, i.e., d_i is inferred if $a_i \in M$ and $a_i \notin N$. Finally, the rules in items (7.)–(9.) guarantee that no counter-models to the $\langle V, \emptyset \rangle$-minimality of M exist.

The stable models of $\mathrm{Tr_V}(\Pi)$ are in a bijective correspondence with the $\langle V, \emptyset \rangle$-minimal models of Π. Furthermore, N is a stable model of $\mathrm{Tr_V}(\Pi)$ if and only if $M = N \cap \mathrm{Hb}(\Pi)$ is a $\langle V, \emptyset \rangle$-minimal model of Π [7]. Composing the translations $\mathrm{Tr_F}(\cdot)$ and $\mathrm{Tr_V}(\cdot)$ we obtain a bijective correspondence between the $\langle V, F \rangle$-minimal models of a program Π and the stable models of the translation $\mathrm{Tr_V}(\mathrm{Tr_F}(\Pi))$. Our approach has been implemented and the tool called CIRC2DLP [11] along with examples and a set of diagnosis benchmarks is available at `http://www.tcs.hut.fi/Software/circ2dlp/`.

Example 1. Consider the program $\Pi = \{f \vee ab.\}$ (a simplified version of Lifschitz's ostrich example [10]). Π has a unique $\langle \{f\}, \emptyset \rangle$-minimal model $M = \{f\}$. The translation $\mathrm{Tr_V}(\Pi)$ for computing $\langle \{f\}, \emptyset \rangle$-minimal models of Π is the following:

1. $f \leftarrow \sim\overline{f}.\ \overline{f} \leftarrow \sim f.\ \overline{ab} \leftarrow \sim ab.$
2. $ab \leftarrow \sim f.$
3. $f^* \vee ab^* \vee u.$
4. $d \vee u.$
5. $u \leftarrow d, \sim ab.\ u \leftarrow ab^* \sim ab.$
6. $u \leftarrow d, ab^*, \sim\overline{ab}.\ u \vee d \vee ab^* \leftarrow \sim\overline{ab}.$
7. $ab^* \leftarrow u.\ f^* \leftarrow u.$
8. $d \leftarrow u.$
9. $\bot \leftarrow \sim u.$

The translation $\mathrm{Tr_V}(\Pi)$ has one stable model $N = \{f, \overline{ab}, f^*, ab^*, d, u\}$ and it holds that $N \cap \mathrm{Hb}(\Pi) = M$.

5 Conclusions

In this paper we have discussed a linear translation from parallel circumscription into disjunctive logic programming such that a bijective correspondence between the $\langle V, F\rangle$-minimal models of a PDLP Π and the stable models of its translation is obtained. This enables the systematic use of varying and fixed atoms in order to develop more compact formulations of problems as disjunctive logic programs.

References

1. M. Cadoli, T. Eiter, and G. Gottlob. An efficient method for eliminating varying predicates from a circumscription. *Artificial Intelligence*, 54(2):397–410, 1992.
2. J. de Kleer and K. Konolige. Eliminating the fixed predicates from a circumscription. *Artificial Intelligence*, 39(3):391–398, July 1989.
3. M. Gelfond and V. Lifschitz. Compiling circumscriptive theories into logic programs. In *Proceedings of the 7th National Conference on Artificial Intelligence*, pages 455–449, St. Paul, MN, August 1988. AAAI Press.
4. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
5. K. Inoue and C. Sakama. Negation as failure in the head. *Journal of Logic Programming*, 35(1):39–78, 1998.
6. T. Janhunen, I. Niemelä, D. Seipel, P. Simons, and J.-H. You. Unfolding partiality and disjunctions in stable model semantics. *ACM Transactions on Computational Logic*, 2005. To appear, see `http://www.acm.org/tocl/accepted.html`.
7. T. Janhunen and E. Oikarinen. Capturing parallel circumscription with disjunctive logic programs. In José Júlio Alferes and João Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence, JELIA'04*, pages 134–146, Lisbon, Portugal, September 2004. Springer-Verlag. LNAI 3229.
8. J. Lee and F. Lin. Loop formulas for circumscription. In D.L. McGuinness and G. Ferguson, editors, *Proceedings of 19th National Conference on Artificial Intelligence*, pages 281–286, San Jose, California, USA, July 2004. The MIT Press.
9. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 2005. To appear, see `http://www.acm.org/tocl/accepted.html`.
10. V. Lifschitz. Computing circumscription. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 121–127, Los Angeles, California, USA, August 1985. Morgan Kaufmann.
11. E. Oikarinen. CIRC2DLP 1.1 — software for translating parallel circumscription into disjunctive logic programming, 2005.
12. T.C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991.
13. R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
14. C. Sakama and K. Inoue. Embedding circumscriptive theories in general disjunctive programs. In *Proceedings of the 3rd International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 344–357. Springer-Verlag, 1995.
15. T. Wakaki and K. Inoue. Compiling prioritized circumscription into answer set programming. In B. Demoen and V. Lifschitz, editors, *Proceedings of the 20th International Conference on Logic Programming*, pages 356–370, Saint-Malo, France, September 2004. Springer-Verlag.