circ2dlp — Translating Circumscription into Disjunctive Logic Programming[⋆]

Emilia Oikarinen[⋆⋆] and Tomi Janhunen

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, Tomi.Janhunen@tkk.fi

1 Introduction

The *stable model semantics* of disjunctive logic programs (DLPs) is based on minimal models [5, 12] which makes atoms appearing in a disjunctive program false by default. This is often desirable from the knowledge representation point of view, but certain domains become awkward to formalize if all atoms are blindly subject to minimization. In contrast to this, *parallel circumscription* [11] provides a refined notion of minimal models as it distinguishes *varying* and *fixed* atoms in addition to those being falsified. This eases the task of knowledge presentation in many cases. For example, it is 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 [1, 6], varying atoms are not fully covered. Earlier approaches either deal with syntactic subclasses of logic programs [4] or have exponential worst-case space complexities [9, 14]. To the contrary, the system circ2dlp described in this article is based on a new *linear* but *non-modular* transformation [8] which enables the use of existing implementations of disjunctive logic programming such as dlv [10] and GnT [7] for the actual search of minimal models.

The rest of this system description is organized as follows. Section 2 concentrates on specifying the output produced by circ2dlp, i.e. the translation presented in [8], on a high level of abstraction. In Section 3, we provide the reader with some instructions how to use the tool in practice. The last section (Section 3) comprises of some preliminary experimental results obtained using circ2dlp together with dlv and GnT. As a benchmark, we use the problem of finding Reiter-style minimal diagnoses [13] for digital circuits. Moreover, we briefly compare the performance of our translation-based approach with another: circum [15] is a system developed for computing *prioritized circumscription* (a generalization of parallel circumscription) using a *generate* and *test* method.

⋆ The research reported in this paper is partially funded by the Academy of Finland (project #211025) and the European Commission (contract IST-FET-2001-37004).

⋆⋆ The support from Helsinki Graduate School in Computer Science and Engineering, Nokia Foundation, and Finnish Cultural Foundation is gratefully acknowledged.

2 Translation-Based Approach

The aim in the following is to briefly describe the translation computed by CIRC2DLP. For that purpose, we give a definition for parallel circumscription in the propositional case. Following the presentation in [8], we formulate the definition in the case of a *positive* DLP Π possessing a Herbrand base $\mathrm{Hb}(\Pi)$. Given a set of *varying* atoms $V \subseteq \mathrm{Hb}(\Pi)$ and a set of *fixed* atoms $F \subseteq \mathrm{Hb}(\Pi)$, the parallel circumscription of Π is characterized by $\langle V, F\rangle$-*minimal* models $M \models \Pi$ each of which is minimal in the following sense: there is no model $N \models \Pi$ such that $N \setminus (V \cup F) \subset M \setminus (V \cup F)$ and $N \cap F = M \cap F$.

The basic idea in the translation-based approach [8] is that the $\langle V, F\rangle$-minimal models of a positive DLP Π can be captured by projecting the stable models of the translation

$$\mathrm{Tr}_{\mathrm{circ2dlp}}(\Pi) = \mathrm{Tr}_{\mathrm{GEN}}(\mathrm{Tr}_{\mathrm{KK}}(\Pi)) \cup \mathrm{Tr}_{\mathrm{EG}}(\mathrm{Tr}_{\mathrm{MIN}}(\mathrm{Tr}_{\mathrm{KK}}(\Pi))) \tag{1}$$

with respect to $\mathrm{Hb}(\Pi)$. The translation given in (1) is based on the following primitives: (i) $\mathrm{Tr}_{\mathrm{KK}}(\cdot)$ removes fixed atoms using a technique proposed in [1], (ii) $\mathrm{Tr}_{\mathrm{GEN}}(\cdot)$ produces a model generator for Π as a DLP, (iii) $\mathrm{Tr}_{\mathrm{MIN}}(\cdot)$ encodes a test for $\langle V, \emptyset\rangle$-minimality as a propositional unsatisfiability problem, and (iv) $\mathrm{Tr}_{\mathrm{EG}}(\cdot)$ implements the required unsatisfiability check using the primitives of DLPs [3]. To summarize the properties of $\mathrm{Tr}_{\mathrm{circ2dlp}}(\Pi)$, the translation is linear in the length of the program $||\Pi||$ and a bijective correspondence between stable models of the translation and the $\langle V, F\rangle$-minimal models of Π is obtained. Further details and the correctness proof of $\mathrm{Tr}_{\mathrm{circ2dlp}}(\Pi)$ can be found in [8].

The model generator $\mathrm{Tr}_{\mathrm{GEN}}(\mathrm{Tr}_{\mathrm{KK}}(\Pi))$ can be enhanced by taking the set of varying atoms V properly into account. Actually, an improved model generator $\mathrm{Tr}_{\mathrm{GEN2}}(\mathrm{Tr}_{\mathrm{KK}}(\Pi))$ is already used in the implementation. The idea is to replace the rules in items 1 and 2 in [8, Definition 6] by the following:

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.

The translation $\mathrm{Tr}_{\mathrm{circ2dlp}}(\Pi)$ optimized in this way is a DLP and thus a valid input for disjunctive solvers implementing the search for stable models [7, 10].

Prioritized circumscription [11] can be translated into parallel circumscription using a scheme proposed by Lifschitz [11]:

$$\mathrm{Circ}(\Pi, P_1 > \ldots > P_k, V) = \bigwedge_{i=1}^{k} \mathrm{Circ}(\Pi, P_i, P_{i+1} \cup \cdots \cup P_k \cup V) \tag{2}$$

where $P_1, \ldots, P_k$ are sets of minimized atoms with decreasing priority, V is the set of varying atoms, and the sets of fixed atoms remain implicit. In our translation-based approach, the equation (2) is understood as a join of k parallel circumscriptions. The respective subtranslations can be concatenated so that $\mathrm{Hb}(\Pi)$ is shared and the new atoms produced by $\mathrm{Tr}_{\mathrm{circ2dlp}}$ remain distinct for each part. It follows that the time complexity is $\mathcal{O}(k \times ||\Pi||)$ in general.

3 Some Instructions for Use

The translator CIRC2DLP has been implemented in C under Linux operating system and is available at `http://www.tcs.hut.fi/Software/circ2dlp/`. The translator takes a DLP Π combined with sets of varying and fixed atoms as input and produces translation $\mathrm{Tr}_{\mathrm{circ2dlp}}(\Pi)$ as output. The input format is the internal format of GNT produced by the front-end LPARSE. Rules with variables can be used, although LPARSE performs an instantiation for the rules. CIRC2DLP produces output compatible with both GNT (default) and DLV.

All atoms are minimized by default, unless explicitly stated to be varying or fixed. Default behaviour can be altered using option `--vary`. Notice that LPARSE might produce *invisible* atoms that have no name in the symbol table. Option `--vary` cannot be applied to programs containing invisible atoms, as the semantics of invisible atoms becomes unclear. CIRC2DLP can also handle programs containing negation. For such programs the translation yields the $\langle V, F\rangle$-minimal models of the Gelfond-Lifschitz reduct of the original program which can be understood as the $\langle V, F\rangle$-stable models of the program.

Command line options for CIRC2DLP are the following:

- `-h` or `--help` – Print a help message.
- `-t` – Print human readable output.
- `--dlv` – Print the output in DLV syntax.
- `--vary` – Vary all atoms by default.
- `--all` – Generate all classical model candidates, using the model generator $\mathrm{Tr}_{\mathrm{GEN}}(\mathrm{Tr}_{\mathrm{KK}}(\Pi))$. Otherwise, $\mathrm{Tr}_{\mathrm{GEN2}}(\mathrm{Tr}_{\mathrm{KK}}(\Pi))$ is used.
- `--version` – Print version information.

For example, all $\langle\{a\}, \emptyset\rangle$-minimal models of a program stored in a file `example.lp` can be computed as follows:

```
lparse --dlp example.lp > example.sm
circ2dlp example.sm -v a | gnt 0
```

or, with DLV,

```
circ2dlp --dlv example.sm -v a | dlv -n=0 --
```

For more examples, see `http://www.tcs.hut.fi/Software/circ2dlp/`.

The translator CIRC2DLP is accompanied by a Perl implementation of Lifschitz's scheme for computing prioritized circumscription called PRIO_CIRC2DLP. For example, the script is used to compute prioritized circumscription $\mathrm{Circ}(\Pi, \{a\} > \{b\}, \emptyset)$ for program Π given in file `example.sm` as follows:

```
prio_circ2dlp example.sm a:b | gnt 0
```

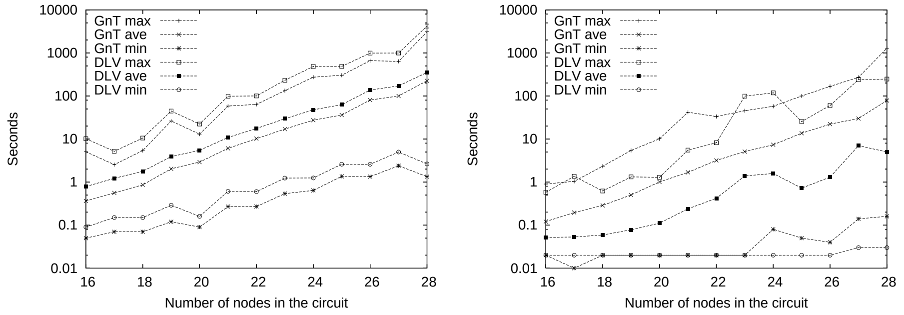

Fig. 1. Computing minimal diagnoses for faulty digital circuits. On the left *all* diagnoses are computed, whereas only *one* diagnosis on the right.

4 Experiments

As a benchmark, we use the problem of finding Reiter-style minimal diagnoses [13] for digital circuits encoded as parallel circumscription. We generate circuits as follows. First, a random tree is generated using the inverse Pruefer algorithm. The leaves of the tree corresponding to the inputs of the digital circuit are assigned random Boolean values. The intermediate nodes are assigned random logical operations corresponding to the intermediate gates of the circuit. The gate at the root node of the tree produces the output of the circuit. The value of the output is calculated and flipped in order to obtain faulty behaviour for the circuit. The number of nodes N in the tree forming the digital circuit varies from 16 to 28. For each number of nodes we generate 100 random instances. These instances are available at `http://www.tcs.hut.fi/Software/circ2dlp/`. Typically an instance has less than ten minimal diagnoses when $N = 28$.

The measured running time is the translation time of CIRC2DLP plus the duration of the search for stable models using GNT or DLV. The actual translation times are negligible, however. We use user+system time of `/usr/bin/time` command in UNIX. All the tests were run under the Debian GNU/Linux 2.4.26 operating system on a AMD Athlon XP 2000+ computer with 1 GB memory.

Results are illustrated in Fig. 1. In the case of finding *all* minimal diagnoses, GNT outperforms DLV, but in the case of finding a *single* minimal diagnosis DLV is superior to GNT in most of the cases. One obvious advantage of our translation-based approach is that it is rather easy to use different solvers and thus gain from their development in the future.

We also compared briefly the performance of our approach with that of the CIRCUM system [15] using some instances of our diagnosis benchmark. Our experiments suggest that in the case of parallel circumscription the running times for the CIRCUM system are one or two orders of magnitude higher than running times for CIRC2DLP+GNT. To compare the systems in the case of prioritized circumscription we used our diagnosis benchmarks and added random priorities for the minimized atoms varying the number of priority classes k. These experiments suggest that our approach is able to compete with CIRCUM when k is small, but as k grows, the quadratic blowup implied by (2) becomes apparent.

There is also a diagnosis front-end in DLV [2], but there are restrictions in the case of minimal diagnoses: the theory has to be non-disjunctive and the abnormality atoms may only appear negatively. This limits the applicability of the front-end so that our diagnosis benchmark cannot be represented naturally.

References

1. J. de Kleer and K. Konolige. Eliminating the fixed predicates from a circumscription. *Artificial Intelligence*, 39(3):391–398, July 1989.
2. T. Eiter, W. Faber, N. Leone, and G. Pfeifer. The diagnosis frontend of the DLV system. *AI Communications*, 12(1-2):99–111, 1999.
3. T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Math. and AI*, 15:289–323, 1995.
4. 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.
5. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
6. K. Inoue and C. Sakama. Negation as failure in the head. *Journal of Logic Programming*, 35(1):39–78, 1998.
7. 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`.
8. T. Janhunen and E. Oikarinen. Capturing parallel circumscription with disjunctive logic programs. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence, JELIA'04*, pages 134–146, Lisbon, Portugal, September 2004. Springer-Verlag. LNAI 3229.
9. J. Lee and F. Lin. Loop formulas for circumscription. In *Proceedings of 19th National Conference on Artificial Intelligence*, pages 281–286, San Jose, California, USA, July 2004. The MIT Press.
10. 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`.
11. 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.
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 *Proceedings of the 20th International Conference on Logic Programming*, pages 356–370, Saint-Malo, France, September 2004. Springer-Verlag.