2009

One More Decidable Class of Finitely Ground Programs

Yuliya Lierler
University of Nebraska at Omaha, ylierler@unomaha.edu

Vladimir Lifschitz
University of Texas at Austin

One More Decidable Class of Finitely Ground Programs

Yuliya Lierler and Vladimir Lifschitz

Department of Computer Sciences, University of Texas at Austin
{yuliya,vl}@cs.utexas.edu

Abstract. When a logic program is processed by an answer set solver, the first task is to generate its instantiation. In a recent paper, Calimeri *et el.* made the idea of efficient instantiation precise for the case of disjunctive programs with function symbols, and introduced the class of "finitely ground" programs that can be efficiently instantiated. Since that class is undecidable, it is important to find its large decidable subsets. In this paper, we introduce such a subset—the class of argument-restricted programs. It includes, in particular, all finite domain programs, ω-restricted programs, and λ-restricted programs.

1 Introduction

When an answer set solver, such as SMODELS[1] or DLV[2], starts processing a logic program Π, the first task is to generate an instantiation of Π—a program without variables that has the same answer sets as Π. In the course of instantiation, the rules of Π are grounded and simplified. Efficient instantiation algorithms expect that each rule of the input program is safe, in the sense that every variable occurring in the rule occurs in the positive part of its body. Some solvers impose stronger restrictions and expect that the given program is ω-restricted [1] or, more generally, λ-restricted [2].

For a program containing function symbols, however, even safety does not guarantee the possibility of instantiating the program efficiently. In fact, a safe program with functions can have infinite answer sets as, for instance, the program

$$\begin{array}{l} p(0) \\ p(f(X)) \leftarrow p(X). \end{array} \tag{1}$$

Such a program cannot be instantiated in a computationally meaningful way. In [3], the idea of efficient (or "intelligent") instantiation is made precise for disjunctive programs with function symbols. Efficient instantiation, as understood in that paper, is applicable to the logic programs that the authors call *finitely ground*. A program without function symbols is finitely ground if and only if it is safe. The program

$$\begin{array}{l} p(0) \\ q(f(X)) \leftarrow p(X) \end{array} \tag{2}$$

[1] LPARSE+SMODELS: `http://www.tcs.hut.fi/Software/smodels/` .
[2] DLV: `http://www.dbai.tuwien.ac.at/proj/dlv/` .

is finitely ground, but program (1) is not. Every finitely ground program has finitely many answer sets, and each of them is finite [3, Corollary 1]. Furthermore, there exists an algorithm for computing the answer sets of an arbitrary finitely ground program [3, Theorem 2].

It appears then that "finitely ground" is a property that a reasonable answer set solver can expect of its input. Unfortunately, the class of finitely ground programs is not decidable [3, Theorem 5]. This fact led the authors to the problem of describing large decidable subclasses of that class. As a step in this direction, they defined a decidable class of "finite domain" programs, and showed that every finite domain program is finitely ground [3, Theorems 6, 7].

In this paper, we introduce another decidable class of finitely ground programs, *argument-restricted* programs, which is a proper superset of the class of finite domain programs. For instance, the program

$$\begin{array}{l} p(f(X)) \leftarrow q(X) \\ q(X) \leftarrow p(X), r(X) \end{array} \tag{3}$$

is argument-restricted, but not finite domain program. The new class is also a superset of λ-restricted programs (and consequently of ω-restricted programs, in view of Theorem 1 from [2]). For instance, the program

$$\begin{array}{l} p(X) \leftarrow q(X) \\ q(X) \leftarrow p(X) \end{array} \tag{4}$$

is argument-restricted (as any safe program without function symbols), but not λ-restricted.

Figure 1 illustrates the relationships between the classes of logic programs mentioned above. The broken line shows the boundary of the important, but undecidable, class of finitely ground programs. In the picture, the class of finite domain programs and the class of λ-restricted programs partially overlap: the former contains program (4), but not (3); the latter contains (3), but not (4).

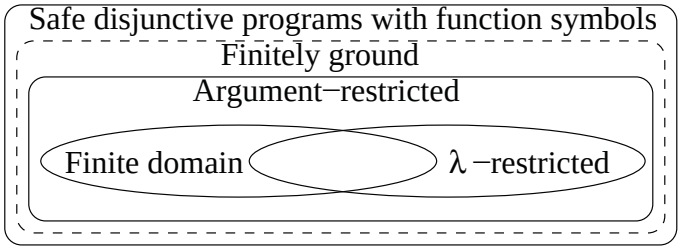

Fig. 1. Classes of logic programs

2 Argument Rankings

We consider disjunctive logic programs—finite sets of rules of the form

$$A_1; \ldots; A_l \leftarrow A_{l+1}, \ldots, A_m,\ \mathit{not}\ A_{m+1}, \ldots,\ \mathit{not}\ A_n \tag{5}$$

($n \geq m \geq l \geq 0$), where each A_i is an atom, possibly containing function symbols. The *positive body* of a rule (5) is the list $A_{l+1}, \ldots, A_m$. A program Π is *safe* if every variable occurring in a rule of Π occurs also in the positive body of that rule. Recall that *grounding* a logic program replaces each rule with all its instances obtained by substituting ground terms, formed from the object and function symbols occurring in the program, for all variables. The *answer sets* of a program are answer sets of the result of its grounding [4].

The definition of an argument ranking below, which is the main definition introduced in this paper, uses the following terminology and notation. For any atom $p(t_1, \ldots, t_n)$, by $p(t_1, \ldots, t_n)^0$ we denote its predicate symbol p, and by $p(t_1, \ldots, t_n)^i$, where $1 \leq i \leq n$, we denote its argument term t_i. As in [3], an *argument* is an expression of the form $p[i]$, where i is one of the argument positions $1, \ldots, n$. Finally, the *depth* of a variable X in a term t that contains X, denoted by $d(X, t)$, is defined recursively, as follows:

$$d(X,t) = \begin{cases} 0, & \text{if } t \text{ is } X, \\ 1 + \max\limits_{i\,:\,t_i \text{ contains } X} d(X, t_i), & \text{if } t \text{ is } f(t_1, \ldots, t_n). \end{cases}$$

An *argument ranking* for a program Π is a function α from arguments to integers such that, for every rule R of Π, every atom A occurring in the head of R, and every variable X occurring in an argument term A^i, the positive body of R contains an atom B such that X occurs in an argument term B^j satisfying the condition

$$\alpha\left(A^0[i]\right) - \alpha\left(B^0[j]\right) \geq d(X, A^i) - d(X, B^j). \tag{6}$$

A safe program is *argument-restricted* if it has an argument ranking.

Example 1. If a safe program Π does not contain function symbols in the heads of rules then it is argument-restricted, because its argument ranking can be defined by $\alpha(p[i]) = 0$ for all arguments $p[i]$. Indeed, the right-hand side of (6) for such a program is nonpositive, because $d(X, A^i) = 0$.

Example 2. Program (1) is not argument-restricted. In fact, any program containing the second rule of (1) is not argument-restricted, because for that rule condition (6) turns into $\alpha(p[1]) - \alpha(p[1]) \geq 1 - 0$.

Example 3. Program (2) is argument-restricted: take $\alpha(p[1]) = 0$, $\alpha(q[1]) = 1$.

Example 4. Program (3) is argument-restricted: take $\alpha(p[1]) = 1$, $\alpha(q[1]) = \alpha(r[1]) = 0$.

Example 5. The one-rule program $p(X, f(X)) \leftarrow p(X, X)$ is argument-restricted: take $\alpha(p[1]) = 0$, $\alpha(p[2]) = 1$.

It is clear that adding the same number to all values of an argument ranking produces another argument ranking for the same program. It follows that any argument-restricted program has an argument ranking with nonnegative values.

3 Properties of Argument-Restricted Programs

Theorem 1 *The set of argument-restricted programs is decidable.*

The easiest proof refers to the fact that the definition of an argument ranking, viewed as a condition on the values of $\alpha(p[i])$, can be encoded in difference logic [5]. A polynomial-time decision method for the set of argument-restricted programs is described in the next section.

The concept of a finitely ground program is defined in [3, Section 3].

Theorem 2 *Every argument-restricted program is finitely ground.*

The concept of a finite domain program is defined in [3, Section 5].

Theorem 3 *Every finite domain program is argument-restricted.*

As mentioned in the introduction, program (3) is a counterexample showing that the converse does not hold. The one-rule program $p(f(X)) \leftarrow p(g(X))$ and the program from Example 5 provide counterexamples as well: they are not finite domain programs, but they are argument-restricted.

The concept of a λ-restricted program is defined in [2, Section 2].

Theorem 4 *Every λ-restricted program is argument-restricted.*

As mentioned in the introduction, program (4) is a counterexample showing that the converse does not hold. The argument-restricted program from Example 5 is not λ-restricted either.

The definition of a λ-restricted program and the definition of an argument-restricted program are similar to each other in the sense that each of them refers to the existence of a number-valued function with certain properties. The difference is that the function is defined on predicate symbols p in the first case, and on arguments $p[i]$ in the second case. We know from Example 5 that the possibility of assigning different values to $p[1]$ and $p[2]$ is essential. Furthermore, the definition of a λ-restricted program does not take into account the depth of nesting of function symbols within a term. If we replace an occurrence of a term—say, $f(X)$— in a λ-restricted program with another term containing the same variables—say, X or $f(f(X))$—the result will be λ-restricted as well.

4 Checking whether a Program Is Argument-Restricted

Recall that the definition of an argument ranking (Section 2) involves a condition on every

(i) rule R of the given program,
(ii) atom A occurring in the head of R,
(iii) argument position i of A, and
(iv) variable X occurring in A^i.

The inequality (6) in that condition can be rewritten as

$$\alpha\left(A^0[i]\right) \geq \alpha\left(B^0[j]\right) + d(X, A^i) - d(X, B^j). \tag{7}$$

For any R, A, i, X satisfying (i)–(iv), by $D_{R,A,i,X}(\alpha)$ we denote the list of the right-hand sides of inequalities (7) for all atoms B in the positive body of R and the argument positions j such that X occurs in B^j. Define the operator Ω on the set U of functions from arguments to nonnegative integers by the formula

$$\Omega(\alpha)(p[i]) = \max\left(\max_{R,A,X\,:\,A^0=p}\left(\min D_{R,A,i,X}(\alpha)\right), 0\right).$$

A function $\alpha \in U$ is an argument ranking for Π iff $\alpha \geq \Omega(\alpha)$.

The operator Ω is monotone. It follows that if Π is argument-restricted then the set of its nonnegative argument rankings has the least element $\alpha_{\min}$, and that $\alpha_{\min} = \Omega^i(0)$ for the smallest i such that $\Omega^{i+1}(0) = \Omega^i(0)$.

On the other hand, we can show that, for any argument-restricted Π, all values of $\alpha_{\min}$ do not exceed the number M defined as the product of the total number of arguments and the largest of the numbers $d(X, t)$ for the terms t occurring in the heads of rules and for the variables X occurring in t.

It follows that we can determine whether Π is argument-restricted by iterating Ω on 0 until

– $\Omega^{i+1}(0) = \Omega^i(0)$ —then $\alpha_{\min}$ is found, or
– one of the values of $\Omega^i(0)$ exceeds M —then Π is not argument-restricted.

Acknowledgements Thanks to Martin Gebser, Tomi Janhunen, Joohyung Lee, Nicola Leone, Ilkka Niemelä, Wanwan Ren, Fangkai Yang for useful discussions. This research was partially supported by the NSF under Grant IIS-0712113.

References

1. Syrjänen, T.: Omega-restricted logic programs. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning. (2001) 267–279
2. Gebser, M., Schaub, T., Thiele, S.: Gringo: A new grounder for answer set programming. In: Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning. (2007) 266–271
3. Calimeri, F., Cozza, S., Ianni, G., Leone, N.: Computable functions in ASP: theory and implementation. In: Proceedings of International Conference on Logic Programming (ICLP). (2008) 407–424
4. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9** (1991) 365–385
5. Mahfoudh, M., Niebert, P., Asarin, E., Maler, O.: A satisfiability checker for difference logic. In: Proceedings of International Conference on the Theory and Applications of Satisfiability Testing (SAT). (2002) 222–230