Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 137 (2005) 5–24
www.elsevier.com/locate/entcs

Elimination of Local Variables from Definite Logic Programs ⋆

Javier Álvez [1] and Paqui Lucio [2]

Departamento de Lenguajes y Sistemas Informáticos
Universidad del País Vasco
San Sebastián, Spain

Abstract

In logic programming, a variable is said to be local if it occurs in a clause body but not in its head atom. It is well-known that local variables are the main cause of inefficiency (sometimes even incompleteness) in negative goal computation. The problem is twofold. First, the negation of a clause body that contains a local variables is not expressible without universal quantification, whereas the abscence of local variables guarantees that universal quantification can be avoided to compute negation. Second, computation of universal quantification is an intrinsically difficult task. In this paper, we introduce an effective method that takes a definite logic program and transforms it into a *local variable free* (definite) program. Source and target programs are equivalent w.r.t. three-valued logical consequences of program completion. In further work, we plan to extend our results to normal logic programs.

Keywords: local variables, logic programming, program transformation.

1 Introduction

Local variables are very often used in logic programs to store intermediate results that are passed from one atom to another in a clause body. It is well-known that local variables cause several problems for solving negative goals, since they give raise to unavoidable universal quantification in the negation of a clause body. Depending on the LP or CLP approach, universal quantification

⋆ This work has been partially supported by Spanish Projects TIC 2001-2476-C03 and TIN2004-079250-C03-03.

[1] Email: jibalgij@si.ehu.es
[2] Email: jiplucap@si.ehu.es

doi:10.1016/j.entcs.2005.01.037

affects simple goals or constrained goals. In the so-called *intensional negation* (cf. [2]) for the LP approach, universal quantification prevents from achieving a complete goal computation mechanism. Afterwards, constructive negation was introduced in [4,5] and extended in [8,16] to a complete and sound operational semantics for the whole class of normal logic programs in the CLP framework. Intensional negation was also extended to CLP in [3] where a complete operational semantics is provided. The computational mechanisms proposed in [3,8,16] deal with universally quantified (constrained) goals that, in general, are not easy to compute in an efficient manner. Besides, the *negation technique* is introduced in [14] and local variable absence is claimed as a sufficient condition for the completeness of the technique.

In this paper, we present an effective transformation method for eliminating local variables from definite logic programs. The underlying aim is to improve the performance of a practical implementation of constructive negation (cf. [1]). Efficiency is achieved because: (1) the negative query is computed w.r.t. an equivalent definite logic program that does not contain any local variable, hence universal quantification is avoided; and (2) the target program is built at compilation time. We would like to remark that the transformed program (without local variables) must only be used to compute negative literals, using the original one for positive literals. Source and target programs are equivalent w.r.t. the standard Clark-Kunen semantics for normal (in particular, definite) logic programs. In further work, we plan to extend our results to normal logic programs.

Our method is unfold/fold-based in the sense that its correctness is given by an unfold/fold transformation sequence. Besides, the transformation relies in a preliminary partition of the argument positions inside the atoms. This partition, called *mode specification*, associates a mode (input/output) to each argument position. Mode specifications are automatically inferred according to the local variables that are going to be eliminated. The mode specification is only used during local variable elimination and it has neither to do with restricting user-goals nor with the dataflow that is assumed by the programmer. Mode analysis and specification is used for several purposes such as compiler optimization, parallel goal-evaluation, etc. (for instance, [7,10]), which are far from the aim of this work. The elimination method requires a previous syntactical normalization of the program with respect to its local variable occurences.

Outline of the paper. In the next section, we give some preliminary definitions. Program normalization is presented in Section 3. The fourth section introduces the notion of mode specification. In Section 5, we show how to eliminate the local variables from a definite program in several phases. Fi-