Simplifying Logic Programs
Under Answer Set Semantics

David Pearce*

Department of Informatics, Statistics and Telematics
Universidad Rey Juan Carlos, Madrid, Spain
d.pearce@escet.urjc.es

Abstract. Now that answer set programming has emerged as a practical tool for knowledge representation and declarative problem solving there has recently been a revival of interest in transformation rules that allow for programs to be simplified and perhaps even reduced to programs of ‘lower’ complexity. Although it has been known for some that there is a maximal monotonic logic, denoted by $N_5$, with the property that its valid (equivalence preserving) inference rules provide valid transformations of programs under answer set semantics, with few exceptions this fact has not really been exploited in the literature. The paper studies some new transformation rules using $N_5$-inference to simplify extended disjunctive logic programs known to be strongly equivalent to programs with nested expressions.

1 Introduction

With the emergence of answer set solvers such as DLV [20], GnT [18], and smodels [33], answer set programming (ASP) now provides a practical and viable environment for tasks of knowledge representation and declarative problem solving. Applications of this paradigm include planning and diagnosis, as exemplified in a prototype decision support system for the space shuttle [2], the management of heterogenous data in information systems, as performed in the INFOMIX project\(^1\), the representation of ontologies in the semantic web allowing for default knowledge and inference, as discussed in [5], as well as compact and fully declarative representations of hard combinatorial problems such as n-Queens, Hamiltonian paths, and so on\(^2\).

Following the rise of ASP as a practical tool, there has recently been a revival of interest in transformation rules that allow for a program to be simplified and perhaps even reduced to a program of ‘lower’ complexity, eg reducing a disjunctive program to a normal program. Recent studies have included [4, 25, 31, 11]. Although it has been known since [27] that there is a maximal monotonic logic, denoted by $N_5$, with the property that all of its valid (equivalence preserving) inference rules provide valid transformations of programs under answer set semantics, with few exceptions this fact has not

---

\(^1\) http://sv.mat.unical.it/infomix/
\(^2\) For these and other examples as well as a general introduction to ASP, see [3].

© Springer-Verlag Berlin Heidelberg 2004
really been exploited in the literature. In this paper we explore several ways in which inference in $N_5$ can be used for program simplification and other computational purposes. The main contributions of the paper, in order of presentation, are as follows. In §2 we give an informal account of how the logic $N_5$ can be employed as tool for program transformations in ASP. §3 provides the logical background and summarises the main known results showing how $N_5$ provides a suitable logical foundation for ASP. In §4 we illustrate how $N_5$ can be used to check that a given transformation rule preserves semantic equivalence of the programs concerned and we discuss new rules for program simplification based on deriving literals (and their negations) from the program in $N_5$. Continuing this theme, in §5 we show how other kinds of $N_5$-derivability may yield computationally useful metatheoretic properties. Some brief remarks on the complexity of these methods follows in §6 and in §7 we conclude with some remarks on related work and on future research topics.

2 Monotonic vs. Nonmonotonic Semantic Transformation Rules

Since 1995, (published as [27]), it has been known that the nonclassical logic of here-and-there with strong negation, $N_5$, provides a suitable logical foundation for the study of programs and theories under answer set semantics. One property of $N_5$ is basic here: answer sets of logic programs correspond to simple kinds of minimal $N_5$-models, called equilibrium models. It was at once apparent that this property could be useful in evaluating putative transformation rules for logic programs, in particular to check the property that a rule is valid under answer set semantics, ie preserves the answer sets of the program being transformed. In particular, any transformation of a program that proceeds according to a valid, given or derived, inference rule of $N_5$ will lead to a logically equivalent program having the same models and therefore the same (minimal) equilibrium models and the same answer sets. It is evident that there are two immediate applications for this property: it may be used to give rather simple proofs that certain known transformation rules are valid for answer set semantics, and it may prove useful in helping to find new rules that preserve the semantics. This was pointed out in [27] but not systematically exploited at the time. More recently this fact was used by others, notably Osorio et al [25], to verify the validity of certain rules such as TAUT, RED-, NONMIN and others considered by Brass and Dix [4]. It is interesting to note that while this shows that answer sets are preserved under any transformations valid in intuitionistic logic, besides some stronger ones, the same is not true of the weaker well-founded semantics, WFS. There are intuitionistically valid transformations that do not preserve the well-founded semantics of a program\(^3\).

In fact it is easy to see that transformations of programs that are valid in $N_5$, ie that take a program $\Pi$ to an $N_5$-equivalent program $\Pi'$, have a still stronger property. Not only are the programs equivalent under answer set semantics, they must be equivalent for all possible extensions $\Pi \cup \Sigma, \Pi' \cup \Sigma$, for the obvious reason that these extended

\(^3\) Consider the program $\Pi$ consisting of two rules (written as logical formulas) $\neg a \rightarrow a; \neg a \rightarrow b$. Since in intuitionistic logic, $\neg a \rightarrow a \vdash \neg a \rightarrow b$, the second rule can be eliminated without loss. However the WFS of the resulting program $\neg a \rightarrow a$ is different from that of $\Pi$. 