Reducing Propositional Theories in Equilibrium Logic to Logic Programs

Pedro Cabalar[1], David Pearce[2], and Agustín Valverde[3]

[1] Dept. of Computation, Univ. of Corunna, Spain
cabalar@dc.fi.udc.es
[2] Dept. of Informatics, Statistics and Telematics,
Univ. Rey Juan Carlos,
(Móstoles, Madrid), Spain
d.pearce@escet.urjc.es
[3] Dept. of Applied Mathematics,
Univ. of Málaga, Spain
a_valverde@ctima.uma.es

Abstract. The paper studies reductions of propositional theories in equilibrium logic to logic programs under answer set semantics. Specifically we are concerned with the question of how to transform an arbitrary set of propositional formulas into an equivalent logic program and what are the complexity constraints on this process. We want the transformed program to be equivalent in a strong sense so that theory parts can be transformed independent of the wider context in which they might be embedded. It was only recently established [1] that propositional theories are indeed equivalent (in a strong sense) to logic programs. Here this result is extended with the following contributions. (i) We show how to effectively obtain an equivalent program starting from an arbitrary theory. (ii) We show that in general there is no polynomial time transformation if we require the resulting program to share precisely the vocabulary or signature of the initial theory. (iii) Extending previous work we show how polynomial transformations can be achieved if one allows the resulting program to contain new atoms. The program obtained is still in a strong sense equivalent to the original theory, and the answer sets of the theory can be retrieved from it.

1 Introduction

Answer set programming (ASP) is fast becoming a well-established environment for declarative programming and AI problem solving, with several implemented systems and advanced prototypes and applications. Though existing answer set solvers differ somewhat in their syntax and capabilities, the language of disjunctive logic programs with two negations, as exemplified in the `DLV` system [11] under essentially the semantics proposed in [7], provides a standard reference point. Many systems support different extensions of the language, either through direct implementation or through reductions to the basic language. For example weight constraints are included in `smodels` [24], while a system called `nlp` [22]

C. Bento, A. Cardoso, and G. Dias (Eds.): EPIA 2005, LNAI 3808, pp. 4–17, 2005.

for compiling nested logic programs is available as a front-end to DLV. Though differently motivated, these two kinds of extensions are actually closely related, since as [6] shows, weight constraints can be represented equivalently by nested programs of a special kind.

Answer set semantics was already generalised and extended to arbitrary propositional theories with two negations in the system of *equilibrium logic*, defined in [17] and further studied in [18,19,20]. Equilibrium logic is based on a simple, minimal model construction in the nonclassical logic of here-and-there (with strong negation), and admits also a natural fixpoint characterisation in the style of nonmonotonic logics. In [20,13] it was shown that answer set semantics for nested programs [12] is also captured by equilibrium models.

While nested logic programs permit arbitrary boolean formulas to appear in the bodies and heads of rules, they do not support embedded implications; so for example one cannot write in nlp a rule with a conditional body, such as

$$p \leftarrow (q \leftarrow r).$$

In fact several authors have suggested the usefulness of embedded implications for knowledge representation (see eg [3,8,23]) but proposals for an adequate semantics have differed. Recently however Ferraris [5] has shown how, by modifying somewhat the definition of answer sets for nested programs, a natural extension for arbitrary propositional theories can be obtained. Though formulated using program reducts, in the style of [7,12], the new definition is also equivalent to that of equilibrium model. Consequently, to understand propositional theories, hence also embedded implications, in terms of answer sets one can apply equally well either equilibrium logic or the new reduct notion of [5]. Furthermore, [5] shows how the important concept of *aggregate* in ASP, understood according to the semantics of [4], can be represented by rules with embedded implications. This provides an important reason for handling arbitrary theories in equilibrium logic and motivates the topic of the present paper.

We are concerned here with the question how to transform a propositional theory in equilibrium logic into an equivalent logic program and what are the complexity constraints on this process. We want the transformed theory to be equivalent in a strong sense so that theory parts can be translated independent of the wider context in which they might be embedded. It was only recently established [1] that propositional theories are indeed equivalent (in a strong sense) to logic programs. The present paper extends this result with the following contributions. (i) We present an alternative reduction method which seems more interesting for computational purposes than the one presented in [1], as it extends the unfolding of nested expressions shown in [12] and generally leads to simpler logic programs. (ii) We show that in general there is no polynomial transformation if we require the resulting program to share precisely the vocabulary or signature of the initial theory. (iii) Extending the work of [15,16] we show how polynomial transformations can be achieved if one allows the resulting program to contain new atoms. The program obtained is still in a strong sense equivalent to the original theory, and the answer sets of the latter can be retrieved from the answer sets of the former.