Artificial Intelligence 246 (2017) 22–33

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Infinitary equilibrium logic and strongly equivalent logic programs ☆

Amelia Harrison [a,*], Vladimir Lifschitz [a,*], David Pearce [b], Agustín Valverde [c]

[a] University of Texas, Austin, TX, USA
[b] Universidad Politécnica de Madrid, Madrid, Spain
[c] University of Málaga, Málaga, Spain

ARTICLE INFO

Article history:
Received 20 January 2016
Received in revised form 9 November 2016
Accepted 8 February 2017
Available online 21 February 2017

Keywords:
Answer set programming
Strong equivalence
Logic of here-and-there

ABSTRACT

Strong equivalence is an important concept in the theory of answer set programming. Informally speaking, two sets of rules are strongly equivalent if they have the same meaning in any context. Equilibrium logic was used to prove that sets of rules expressed as propositional formulas are strongly equivalent if and only if they are equivalent in the logic of here-and-there. We extend this line of work to formulas with infinitely long conjunctions and disjunctions, show that the infinitary logic of here-and-there characterizes strong equivalence of infinitary formulas, and give an axiomatization of that logic. This is useful because of the relationship between infinitary formulas and logic programs with local variables.

1. Introduction

Answer set programming (ASP) is a form of declarative programming based on the stable model semantics of logic programs [1–7]. The concept of strong equivalence plays an important role in the theory of ASP. Informally speaking, two sets of rules are strongly equivalent if they have the same meaning in any context.

Compare, for instance, the rules

$$q(X, Z) \leftarrow q(X, Y),\ q(Y, Z),\ p(X),\ p(Y),\ p(Z) \tag{1}$$

and

$$\leftarrow q(X, Y),\ q(Y, Z),\ not\ q(X, Z),\ p(X),\ p(Y),\ p(Z). \tag{2}$$

Both rules express the idea that relation q is transitive on domain p. But in many contexts these rules do not have the same meaning: the effect of adding (1) to a logic program describing p and q is, in general, not the same as the effect of adding (2). The first rule allows us to derive new facts about q; adding it to a program turns relation q into its transitive closure. The second rule is a constraint; adding it weeds out the stable models in which q is not transitive.

☆ This paper is an invited revision of a paper which first appeared at the 13th International Conference on Logic Programming and Non-monotonic Reasoning.

* Corresponding author.
E-mail addresses: ameliaj@cs.utexas.edu (A. Harrison), vl@cs.utexas.edu (V. Lifschitz), david.pearce@upm.es (D. Pearce), a_valverde@ctima.uma.es (A. Valverde).

http://dx.doi.org/10.1016/j.artint.2017.02.002
0004-3702/

The situation is different, however, if the program to which we add rules (1) and (2) contains the choice rule

$$\{q(X,Y)\} \leftarrow p(X),\ p(Y) \tag{3}$$

("for any X, Y from p, decide arbitrarily whether to include $q(X,Y)$ in the stable model"). The set consisting of rules (1) and (3) is strongly equivalent to the set consisting of (2) and (3). Consequently, in the presence of choice rule (3), the program obtained by adding (1) has the same stable models as the program obtained by adding (2).

According to Lifschitz et al. [8], strong equivalence is closely related to the 3-valued logic called the logic of here-and-there, which was introduced by Arend Heyting [9] long before the invention of computer programming.[1] Consider the ground instances of rules (1)–(3):

$$\begin{aligned} q(t_1,t_3) &\leftarrow q(t_1,t_2),\ q(t_2,t_3),\ p(t_1),\ p(t_2),\ p(t_3),\\ &\leftarrow q(t_1,t_2),\ q(t_2,t_3),\ \mathit{not}\ q(t_1,t_3),\ p(t_1),\ p(t_2),\ p(t_3),\\ \{q(t_1,t_2)\} &\leftarrow p(t_1),\ p(t_2) \end{aligned}$$

(t_1, t_2, t_3 are arbitrary ground terms) and rewrite these ground rules as propositional combinations of ground atoms in the following way:

$$q(t_1,t_2) \wedge q(t_2,t_3) \wedge p(t_1) \wedge p(t_2) \wedge p(t_3) \rightarrow q(t_1,t_3), \tag{4}$$

$$\neg(q(t_1,t_2) \wedge q(t_2,t_3) \wedge \neg q(t_1,t_3) \wedge p(t_1) \wedge p(t_2) \wedge p(t_3)), \tag{5}$$

$$p(t_1) \wedge p(t_2) \rightarrow q(t_1,t_2) \vee \neg q(t_1,t_2). \tag{6}$$

Formulas (4) and (5) are equivalent to each other in classical logic. But this fact cannot be established in the logic of here-and-there, which is weaker than classical logic. Formula (6) is a tautology; this fact cannot be established in the logic of here-and-there either. On the other hand, the equivalence between the set consisting of formulas of forms (4) and (6) and the set consisting of formulas of forms (5) and (6) can be proved even in this weaker logic. This example illustrates a general fact: two sets of rules written as propositional formulas are strongly equivalent if and only if they are equivalent in the logic of here-and-there [8, Theorem 1].

In view of this relationship, proving strong equivalence can be often reduced to reasoning in a system of axioms and inference rules that is sound and complete with respect to the logic of here-and-there. Such formal systems have been known for a long time; see Section 5.1.

The proof of the theorem relating strong equivalence to the logic of here-and-there is based on the characterization of stable models in terms of equilibrium logic [10]—a nonmonotonic counterpart of the logic of here-and-there.

The statement of the theorem is not restricted to finite sets of formulas. This is important because a single rule with variables has infinitely many ground instances if we allow function symbols (or symbols for arbitrary integers) in ground terms. But some rules found in ASP programs can be represented by sets of propositional formulas only if we allow formulas themselves to be infinite; infinite sets of finite formulas do not suffice. Consider, for instance, the rule

$$q \leftarrow \mathit{count}\{X : p(X)\} = 0. \tag{7}$$

The aggregate expression in the body means, informally speaking, that set p is empty. This rule can be thought of as an implication with an infinite conjunction in the body:

$$\bigwedge_t \neg p(t) \rightarrow q.$$

Here t ranges over ground terms. The need for infinite conjunctions and disjunctions is common when rules contain local variables, such as X in the example above. Many ASP programs, in particular many programs in the input language of the grounder GRINGO and its subset, the ASP Core language [11], can be represented by formulas of this type [12].

In many cases, first-order formulas can also be used to capture the meaning of ASP programs. For example, rule (7) can be represented using the first-order formula

$$\forall x \neg p(x) \rightarrow q.$$

But possibilities of this approach are more limited. For instance, if *count* in (7) is replaced with *sum*, or 0 is replaced by a variable, the resulting rule cannot be represented using a first-order formula.

In this paper, on the basis of the definition of a stable model for infinitary propositional formulas proposed by Miroslaw Truszczyński [13], we extend to such formulas some definitions and theorems of the theory of strong equivalence and equilibrium logic. Our goals are

[1] The name "here-and-there" is appropriate in view of the fact that this logic can be described in terms of Kripke frames with two worlds, "Here" and "There." It is known also as "the logic of present and future" or "the Smetanich logic."