Yet Another Characterization of Strong Equivalence*

Alexander Bochman[1] and Vladimir Lifschitz[2]

1 **Holon Institute of Technology, Israel**
2 **University of Texas at Austin, USA**

Abstract

Strong equivalence of disjunctive logic programs is characterized here by a calculus that operates with syntactically simple formulas.

1998 ACM Subject Classification D.1.6 Logic Programming, I.2.3 Deduction and Theorem Proving

Keywords and phrases Strong equivalence, logic program

Digital Object Identifier 10.4230/LIPIcs.ICLP.2011.11

1 Introduction

Logic programs Π_1 and Π_2 are said to be strongly equivalent to each other if, for every logic program Π, the program $\Pi_1 \cup \Pi$ has the same stable models as $\Pi_2 \cup \Pi$ [4]. The study of strong equivalence is important because we learn from it how one can simplify a part of a logic program without looking at the rest of it. Characterizations of strong equivalence of logic programs that allow us to establish it more easily than by using the definition directly are given in [4], [6], and [7].

According to the main theorem of the first of these papers, grounded programs are strongly equivalent to each other iff the equivalence between them can be proved in the logic of here-and-there HT—the extension of intuitionistic propositional logic obtained by adding to it the axiom schema

$$F \vee (F \to G) \vee \neg G. \tag{1}$$

This statement assumes that grounded rules are viewed as alternative notation for propositional formulas. Specifically, a disjunctive rule

$$A_1; \dots; A_k; not\ A_{k+1}; \dots; not\ A_l \leftarrow A_{l+1}, \dots, A_m, not\ A_{m+1}, \dots, not\ A_n \tag{2}$$

$(n \geq m \geq l \geq k \geq 0)$, where each A_i is an atom, is identified with the propositional formula

$$A_{l+1} \wedge \dots \wedge A_m \wedge \neg A_{m+1} \wedge \dots \wedge \neg A_n \to A_1 \vee \dots \vee A_k \vee \neg A_{k+1} \vee \dots \vee \neg A_l. \tag{3}$$

In the special case when each rule of the program has the form (2) without negation in head ($l = k$), strong equivalence can be characterized by a calculus that operates with such rules directly, without rewriting them as propositional formulas [8]. This fact shows that

* This work was partially supported by the National Science Foundation under Grant IIS-0712113.

Technical Communications of the 27th International Conference on Logic Programming (ICLP'11).
Editors: John P. Gallagher, Michael Gelfond; pp. 11–15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

under some conditions strong equivalence can be described in terms of derivations that do not involve syntactically complex expressions, such as (1).

In this note we show that results of [1, Chapter 5] give us, implicitly, a calculus similar to the one proposed in [8], slightly more general (negation in the head is allowed) and slightly simpler (an inference rule with many premises is replaced by a rule with one premise). In this modification of the calculus from [8], derivable objects are "flat implications"—arbitrary formulas of the form (3). Negation in the head is important because it is needed to encode the choice construct, frequently used in answer set programming [3]. For instance, the choice rule

$$\{p\} \leftarrow q, \mathit{not}\ r$$

can be thought of as shorthand for

$$p; \mathit{not}\ p \leftarrow q, \mathit{not}\ r.$$

2 Calculus of Flat Implications

A *flat implication* is a propositional formula of the form $C \to D$, where C is a conjunction of literals (possibly the empty conjunction $\top$), and D is a disjunction of literals (possibly the empty disjunction $\bot$).

In the description of the calculus of flat implications *CFI* below, A is an atom; L is a literal; C, C_1, C_2 are conjunctions of literals; D, D_1, D_2 are disjunctions of literals; N is a disjunction of negative literals.

The calculus consists of two axiom schemas

$$A \to A, \tag{4}$$

$$A \wedge \neg A \to \bot \tag{5}$$

and three inference rules: *cut*

$$\frac{C_1 \to D_1 \vee L \qquad L \wedge C_2 \to D_2}{C_1 \wedge C_2 \to D_1 \vee D_2},$$

regularity

$$\frac{A \wedge C \to N}{C \to N \vee \neg A},$$

and the *structural rule*

$$\frac{C \to D}{C_1 \to D_1}$$

where each member of C is a member of C_1
and each member of D is a member of D_1.

Theorem. *A flat implication I is derivable from a set Π of flat implications in CFI iff I is derivable from Π in HT.*

In Section 4 we will show that this theorem is essentially a restatement of [1, Theorem 5.36].

Corollary. *For any sets Π_1, Π_2 of flat implications, the following conditions are equivalent:*

- *Π_1 is strongly equivalent to Π_2,*

- *in the calculus of flat implications, each element of* Π_1 *can be derived from* Π_2*, and each element of* Π_2 *can be derived from* Π_1.

The main feature of *CFI* that distinguishes it from the calculus proposed in [8] is the regularity rule, which takes advantage of the availability of negation in the heads of rules.

3 Examples

Example 1. We would like to verify that in the presence of the choice rule $\{p\}$, the rule $p \leftarrow q$ can be replaced by the constraint $\leftarrow q, not\ p$. In other words, we want to show that the program

$$\begin{array}{l} \{p\} \\ p \leftarrow q \end{array}$$

is strongly equivalent to

$$\begin{array}{l} \{p\} \\ \leftarrow q, not\ p. \end{array}$$

According to the corollary above, it is sufficient to derive in the calculus of flat implications

(a) $q \wedge \neg p \rightarrow \bot$ from $q \rightarrow p$;
(b) $q \rightarrow p$ from $\top \rightarrow p \vee \neg p$ and $q \wedge \neg p \rightarrow \bot$.

Part (a):

1. $q \rightarrow p$ (assumption).
2. $p \wedge \neg p \rightarrow \bot$ (axiom).
3. $q \wedge \neg p \rightarrow \bot$ (by cut from 1 and 2).

Part (b):

1. $\top \rightarrow p \vee \neg p$ (assumption).
2. $q \wedge \neg p \rightarrow \bot$ (assumption).
3. $\neg p \wedge q \rightarrow \bot$ (by the structural rule from 2).
4. $q \rightarrow p$ (by cut from 1 and 3).

Example 2. We would like to verify that the disjunctive program

$$\begin{array}{l} p; q \\ \leftarrow p, q \end{array}$$

is strongly equivalent to the nondisjunctive program

$$\begin{array}{rl} p & \leftarrow not\ q \\ q & \leftarrow not\ p \\ & \leftarrow p, q. \end{array}$$

It is sufficient to derive in the calculus of flat implications

(a) $\top \rightarrow p \vee q$ from the formulas

$$\neg q \rightarrow p, \quad \neg p \rightarrow q, \quad p \wedge q \rightarrow \bot;$$

(b) $\neg q \rightarrow p$ and $\neg p \rightarrow q$ from the formulas

$$\top \rightarrow p \vee q, \quad p \wedge q \rightarrow \bot.$$

Part (a):

1. $p \wedge q \to \bot$ (assumption).
2. $q \to \neg p$ (by regularity from 1).
3. $\top \to \neg p \vee \neg q$ (by regularity from 2).
4. $\neg q \to p$ (assumption).
5. $\top \to \neg p \vee p$ (by cut from 3 and 4).
6. $\top \to p \vee \neg p$ (by the structural rule from 5).
7. $\neg p \to q$ (assumption).
8. $\top \to p \vee q$ (by cut from 6 and 7).

Part (b):

1. $\top \to p \vee q$ (assumption).
2. $q \wedge \neg q \to \bot$ (axiom).
3. $\neg q \to p$ (by cut from 1 and 2).

The derivation of $\neg p \to q$ is similar.

4 Proof of the Theorem

According to [1], a *bisequent* is an expression of the form

$$a : b \Vdash c : d \tag{6}$$

where a, b, c, d are finite sets of atoms. Bisequents can be thought of as flat implications in disguise if we agree to identify (6) with the formula

$$\bigwedge_{A \in a} A \wedge \bigwedge_{A \in b} \neg A \;\to\; \bigvee_{A \in c} A \vee \bigvee_{A \in d} \neg A.$$

From [1, Proposition 5.84] we see that, given this convention, stable models of a set Π of flat implications are identical to the extensions of Π in the sense of [1, Definitions 5.7 and 5.8].

The characterization of strong-extension equivalence of bisequent theories given by [1, Theorem 5.36] provides a characterization of strong equivalence of sets of flat implications in terms of a calculus that is almost identical to *CFI*. The axioms of that calculus are our axioms (4) and (5) (called in the book positive reflexivity and consistency, see [1, Definitions 3.1 and 3.8]) plus the axiom schema

$$\neg A \to \neg A \tag{7}$$

(negative reflexivity, see [1, Definition 3.1]). Its inference rules are the inference rules of *CFI* (monotonicity, positive cut, negative cut, and C-regularity, see [1, Definition 5.7 and Section 3.2.5]). It remains to observe that (7) can be derived from (5) by one application of the regularity rule.

The "only if" part of the theorem can be proved also by noting that all postulates of *CFI* can be justified in *HT*. In fact, the axioms, the cut rule, and the structural rule are even intuitionistically acceptable. As to the regularity rule, its conclusion can be intuitionistically derived from its premise and the weak excluded middle axiom $\neg A \vee \neg\neg A$; the latter is provable in *HT* (in the axiom schema (1), take A as F and $\neg A$ as G). This line of reasoning shows, incidentally, that on the level of flat implications the logic of here-and-there does not differ from the logic of the weak excluded middle *WEM*—a fact known from [2].

5 Conclusion

There is a certain degree of freedom when we decide which monotonic logic can be viewed as the basis of the stable model semantics of disjunctive logic programs. From the results of [4] and [2] we see that each of the systems *HT* and *WEM* can play this role; the theorem presented in this note shows that *CFI* would do as well.

In [5], the theorem from [4] is extended to logic programs with variables and to a first-order version of *HT*. It would be interesting to extend the property of *CFI* proved above in a similar way.

6 Acknoweldgements

We are grateful to Fangzhen Lin and to the anonymous referees for useful comments.

References

1 Alexander Bochman. *Explanatory nonmonotonic reasoning*. World Scientific, 2005.
2 Dick De Jongh and Lex Hendriks. Characterization of strongly equivalent logic programs in intermediate logics. *Theory and Practice of Logic Programming*, 3:259–270, 2003.
3 Paolo Ferraris and Vladimir Lifschitz. Weight constraints as nested expressions. *Theory and Practice of Logic Programming*, 5:45–74, 2005.
4 Vladimir Lifschitz, David Pearce, and Agustin Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.
5 Vladimir Lifschitz, David Pearce, and Agustin Valverde. A characterization of strong equivalence for logic programs with variables. In *Procedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 188–200, 2007.
6 Fangzhen Lin. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 170–176, 2002.
7 Hudson Turner. Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3(4,5):609–622, 2003.
8 Ka-Shu Wong. Sound and complete inference rules for SE-consequence. *Journal of Artificial Intelligence Research*, 31:205–216, 2008.