Checking Strong Equivalence with Duplication-Free Tableaux[⋆]

A. Avellone[1], S. Costantini[2], G. Fiorino[3], U. Moscato[1] and A. Provetti[4]

[1] Dipartimento di Metodi Quantitativi per l'Economia, Università Milano-Bicocca,Piazza dell'Ateneo Nuovo, 1, 20126 Milano, Italy,
{alessandro.avellone,ugo.moscato}@unimib.it,
http://www.dimequant.unimib.it/

[2] Dipartimento d'Informatica, Università di L'Aquila, via Vetoio Loc. Coppito, L'Aquila I-67010 Italy, stefcost@di.univaq.it, http://costantini.di.univaq.it/

[3] Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39, 20135 Milano, Italy,
fiorino@dsi.unimi.it

[4] Dipartimento di Fisica, Università di Messina, Messina I-98166 Italy,
ale@unime.it, http://ale.unime.it/

Abstract. In Answer Set Programming, Lifschitz, Pearce and Valverde have defined Strong equivalence as follows: Π_1 and Π_2 are *strongly equivalent* if for every program Π, $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ are equivalent, i.e., have the same answer sets. A logical characterization of Strong Equivalence is provided by the same authors via i) translation of programs into a classical signature (creating, say, $c(\Pi_1)$ and $c(\Pi_2)$) and checking equivalence of so-obtained formulae w.r.t. to the Logic of Here-and-There. Here-and-There is an intermediate logic only minimally weaker than classical logic. Indeed Strong Equivalence can be checked with classical logic model checkers but only at the cost of introducing extra atoms. We remain within Lifschitz et al. logical framework and describe an optimal tableaux system for Here-and-There. Such Tableau is derived from a similar, but weaker system introduced by some of the authors for modal intuitionistic logic. Checking strong equivalence can therefore be done in a very concise form without introducing extra atoms and without duplications in proofs.

1 Introduction

Answer Set Programming (ASP) is a branch of Logic Programming based on the Stable Models and Answer Sets declarative semantics of [9] and

[⋆] Work supported by i) MIUR COFIN project *Aggregate- and number-reasoning for computing: from decision algorithms to constraint programming with multisets, sets, and maps* and by ii) Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-37004 WASP project.

[10]. Essentially, the answer sets semantics relates the negation-as failure operator *not* to the notion of consistent assumption in Reiter's default logic.

Differently from standard Logic Programming, where program statements designate properties of a first-class object which is to be computed, Answer Sets Programming is based on the understanding of program statements as constraints on a set of atoms that encode a solution to the problem at hand. In general, a program may admit *zero,* one or more answer sets (each one encodes an alternative solution). The reader may refer to Marek and Truszczyński's survey [13] for an introduction to ASP visá-vis traditional Logic Programming and classical propositional logic. In this article the standard definitions of (propositional) logic program (or, equivalently, extensions of *DATALOG*) are assumed. Also, the standard definitions of Well-founded [21], Stable [9] and Answer Sets semantics [10] are assumed.

1.1 Introduction to Strong Equivalence

In their seminal work, Lifschitz et al. [12] have defined Strong Equivalence (SE), argued for its importance and given a formal method by which SE can be checked. The key points of their work can be summarized as follows. First, Strong Equivalence is defined.

Definition 1. *(from [12])*
Programs Π_1 is strongly equivalent to program Π_2 if, for every program Π, $\Pi_1 \cup \Pi$ is equivalent to $\Pi_2 \cup \Pi$.

Second, the importance of strong equivalence from the point of view of program development is argued for.

> 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. [ibid.]

On this issue, we would like to add that advances in Strong Equivalence will enable the development of a Software Engineering of ASP, based on *safe* combinations of program modules, which clearly helps in writing a large program.

Finally, Lifschitz et al. provide a formal characterization of strongly equivalent logic programs by means of the logic of Here-and-There (**HT**) wherein [16] had characterized stable models by a distinct class of models called *equilibrium models.*

Theorem 1. *(rephrases Theorem 1 of [12])*
Program Π_1 is strongly equivalent to program Π_2 if and only if they are equivalent[1] *in the sense of HT logic.*

As a result, strong equivalence can be checked by checking HT-equivalence, which can be done, for instance, by means of Modal Tableaux theorem provers.

Example 1. Programs
$\pi_1 = \{p \leftarrow \mathit{not}\ p, \mathit{not}\ q.\ p \leftarrow q.\}$ and $\pi_1 = \{p \leftarrow \mathit{not}\ p.\ p \leftarrow q.\}$ are strongly equivalent. This fact is witnessed by the following **HT** theorem:

$$\models_{HT} [(\neg q \wedge \neg p \rightarrow p) \wedge (q \rightarrow p)] \equiv [(\neg p \rightarrow p) \wedge (q \rightarrow p)]$$

Notice that the **HT** formula is extracted from π_1 and π_2 by simply replacing the Logic Programming connectives with classical ones. The Clark's completion of programs is not used at all.

1.2 HT from the point of view of intermediate logic

In this paper we are interested in propositional *Here-and-There* Logic, which can be axiomatized by adding to any axiom system for Intuitionistic Logic **Int** the axiom schema

$$((\neg q \rightarrow p) \rightarrow (((p \rightarrow q) \rightarrow p) \rightarrow p))$$

Equivalently, **HT** may be axiomatized by adding to **Int** the two axiom-schemes

$$((p \rightarrow q) \vee (q \rightarrow p))$$

$$(p \vee (p \rightarrow q \vee \neg q))$$

Clearly, in both cases adding all the possible instantiations of p and q appearing in the schema may burden the theorem prover with several scarcely-needed axioms. In the next Section we will show an alternative characterization based on introducing truth-modalities (*signs*) as prefixes to the formulae.

[1] Logic programs are translated into HT formulae by replacing connectives with their classical counterparts. In particular, Gelfond-Lifschitz's $\leftarrow$, which is not truth-valued, gets substituted by the classical implication, denoted $\rightarrow$.

2 Basic definitions

In this section we give notions and notation we will use in the rest of the article. A detailed presentation of all notions regarding intermediate logics and Kripke models can be found in [7] and [3].

Given a finite set[2] of propositional variables (i.e., atoms) and the connectives $\neg, \wedge, \vee, \rightarrow$, a *well formed formula (wff* for short) is defined as usual. Given a wff A, we say that $\neg A$ is a *negated wff.* We use the term *atom* as synonym of propositional variable.

In the sequel **Int** denotes both an Hilbert-style calculus for Intuitionistic Propositional Logic and the set of intuitionistically valid wffs.

A well known semantical characterization of **HT** is by *Kripke models.* A Kripke model is a triple $\underline{K} = \langle P, \leq, \Vdash \rangle$, where $\langle P, \leq \rangle$ is a partial ordered set and $\Vdash$ is the *forcing relation*, defined between elements of P and propositional variables with the property that, for every $\Gamma, \Delta \in P$ such that $\Gamma \leq \Delta$, and every propositional variable p, if $\Gamma \Vdash p$ then $\Delta \Vdash p$.

We call *root of* $\underline{K} = \langle P, \leq, \Vdash \rangle$ an element Υ (if it exists) such that, for every $\Gamma \in P$, $\Upsilon \leq \Gamma$.

HT is semantically characterized by the class of rooted Kripke models $\underline{K} = \langle P, \leq, \Vdash \rangle$ such that $|P| \leq 2$.

The *forcing relation* is extended to the wffs as follows:

- $\Gamma \Vdash A \wedge B$ iff $\Gamma \Vdash A$ and $\Gamma \Vdash B$;
- $\Gamma \Vdash A \vee B$ iff $\Gamma \Vdash A$ or $\Gamma \Vdash B$;
- $\Gamma \Vdash A \rightarrow B$ iff, for every $\Delta \in P$ such that $\Gamma \leq \Delta$, $\Delta \nVdash A$ or $\Delta \Vdash B$;
- $\Gamma \Vdash \neg A$ iff for every $\Delta \in P$ such that $\Gamma \leq \Delta$, $\Delta \nVdash A$.

Starting from the definition above, it is easy to prove that if $\Gamma \Vdash A$ and $\Gamma \leq \Delta$ then $\Delta \Vdash A$.

Let Γ be a world of P and let A be a wff, if $\Gamma \Vdash A$ we say that A *is forced in* Γ (or in a world of $\underline{K}$). A wff A is *valid* in a *model* $\underline{K} = \langle P, \leq, \Vdash \rangle$ if $\Gamma \Vdash A$ for all $\Gamma \in P$.

In Section 3 we present the tableau calculus **T** for **HT** given in [1], whose object language is built on the set of signs $\{\mathbf{T}, \mathbf{F}, \mathbf{F_c}, \mathbf{T_{cl}}\}$ and on the set of wffs. Every member of the object language is a *signed wff* (*swff* for short) whose syntax is $\mathcal{S}A$, with $\mathcal{S} \in \{\mathbf{T}, \mathbf{F}, \mathbf{F_c}, \mathbf{T_{cl}}\}$ and A wff.

The *length* of a wff A (respectively swff $\mathcal{S}A$), denoted by $|A|$ (respectively $|\mathcal{S}A|$), is the number of symbols in A (respectively the number of symbols in A plus one). The length of a set S of wffs or swffs, denoted by

[2] Since ASP programs are finite, in this article we only need to assume the existence of finitely many propositional variables (atoms).

$|S|$, is the sum of the lengths of its elements. Given two wffs or two swffs A, B with $A \equiv B$ we mean that A and B are syntactically identical.

Finally, we introduce a *complexity* measure for wff, swff and sets of swff.

Definition 2. *1. The* degree of a wff *A, denoted by* $\deg(A)$*, is defined as follows: if A is a propositional variable then* $\deg(A) = 0$*; if $A \equiv B \to C$, $A \equiv B \wedge C$ or $A \equiv B \vee C$ then* $\deg(A) = \deg(B) + \deg(C) + 1$*, if $A \equiv \neg B$ then* $\deg(A) = \deg(B) + 1$*.*

2. The degree of a swff *$\mathcal{S}A$, denoted by* $\deg(\mathcal{S}A)$*, coincides with the degree of A.*

3. The degree of a set *S* of swffs*, denoted by* $\deg(S)$*, is the sum of the degrees of the swffs occurring in S.*

3 The calculus for HT

The duplication-free tableau calculus for **HT** given in [1] uses the signs $\mathbf{T}$, $\mathbf{F}$, $\mathbf{F_c}$ and $\mathbf{T_{cl}}$. The meaning of these signs is explained in terms of *realizability* as follows: given a model $\underline{K} = \langle P, \leq, \Vdash \rangle$ and a swff H, we say that an element $\beta \in P$ *realizes* H, and we write $\beta \rhd H$, if (according to the structure of H) the following conditions hold:

- If $H \equiv \mathbf{T}A$, then $\beta \Vdash A$;
- If $H \equiv \mathbf{F}A$, then $\beta \nVdash A$;
- If $H \equiv \mathbf{F_c}A$, then $\beta \Vdash \neg A$;
- If $H \equiv \mathbf{T_{cl}}A$, then $\beta \Vdash \neg\neg A$.

The calculus T consists of the rules in Table 1 and a set S is contradictory if one of the following conditions holds:

- $\mathbf{T}A \in S$ and $\mathbf{F}A \in S$;
- $\mathbf{T}A \in S$ and $\mathbf{F_c}A \in S$;
- $\mathbf{T_{cl}}A \in S$ and $\mathbf{F_c}A \in S$.

It is immediate to verify:

Proposition 1. *If a set of swff's is contradictory, then it is not realizable.*

A world Γ of a model $\underline{K}$ *realizes* a set S of swff's (and we write $\Gamma \rhd S$) iff Γ *realizes* every swff in S. A set S of swff's *realizable* iff there is a world Γ of model $\underline{K}$ such that $\Gamma \rhd S$.

$$\frac{S, \mathbf{T}(A \wedge B)}{S, \mathbf{T}A, \mathbf{T}B}\,\mathbf{T}\wedge \qquad \frac{S, \mathbf{T_{cl}}(A \wedge B)}{S, \mathbf{T_{cl}}A, \mathbf{T_{cl}}B}\,\mathbf{T_{cl}}\wedge$$

$$\frac{S, \mathbf{T}(A \vee B)}{S, \mathbf{T}A / S, \mathbf{T}B}\,\mathbf{T}\vee \qquad \frac{S, \mathbf{T_{cl}}(A \vee B)}{S, \mathbf{T_{cl}}A / S, \mathbf{T_{cl}}B}\,\mathbf{T_{cl}}\vee$$

$$\frac{S, \mathbf{T}(A \to B)}{S, \mathbf{F}A, \mathbf{T_{cl}}B / S, \mathbf{T}B / S, \mathbf{F_c}A}\,\mathbf{T}\to \qquad \frac{S, \mathbf{T_{cl}}(A \to B)}{S, \mathbf{T_{cl}}B / S, \mathbf{F_c}A}\,\mathbf{T_{cl}}\to$$

$$\frac{S, \mathbf{T}\neg A}{S, \mathbf{F_c}A}\,\mathbf{T}\neg \qquad \frac{S, \mathbf{T_{cl}}\neg A}{S, \mathbf{F_c}A}\,\mathbf{T_{cl}}\neg$$

$$\frac{S, \mathbf{F}(A \wedge B)}{S, \mathbf{F}A / S, \mathbf{F}B}\,\mathbf{F}\wedge \qquad \frac{S, \mathbf{F_c}(A \wedge B)}{S, \mathbf{F_c}A / S, \mathbf{F_c}B}\,\mathbf{F_c}\wedge$$

$$\frac{S, \mathbf{F}(A \vee B)}{S, \mathbf{F}A, \mathbf{F}B}\,\mathbf{F}\vee \qquad \frac{S, \mathbf{F_c}(A \vee B)}{S, \mathbf{F_c}A, \mathbf{F_c}B}\,\mathbf{F_c}\vee$$

$$\frac{S, \mathbf{F}(A \to B)}{S, \mathbf{T}A, \mathbf{F}B \ / \ S, \mathbf{T_{cl}}A, \mathbf{F_c}B}\,\mathbf{F}\to \qquad \frac{S, \mathbf{F_c}(A \to B)}{S, \mathbf{T_{cl}}A, \mathbf{F_c}B}\,\mathbf{F_c}\to$$

$$\frac{S, \mathbf{F}\neg A}{S, \mathbf{T_{cl}}A}\,\mathbf{F}\neg \qquad \frac{S, \mathbf{F_c}\neg A}{S, \mathbf{T_{cl}}A}\,\mathbf{F_c}\neg$$

Table 1. Tableau calculus for **HT**

3.1 Describing Proofs structure

A *configuration* is any finite sequence $S_1|\dots|S_n$ (with $n \geq 1$), where every S_j is a set of swff's; a configuration is realizable iff at least a S_j is realizable; we refer to S_j as an *element* of $S_1|\dots|S_n$.

A *proof table* is a finite sequence of configurations $\mathcal{C}_1, \dots, \mathcal{C}_n$, where the configuration $\mathcal{C}_{i+1}$ is obtained from $\mathcal{C}_i = S_1|\dots|S_k$ by applying to each *non-contradictory* element of $\mathcal{C}_i$ a rule of the calculus and taking every contradictory element of $\mathcal{C}_i$ in $\mathcal{C}_{i+1}$. Moreover, a proof table is *closed* iff all the sets S_j in its final configuration are *contradictory.* Finally, the *depth* of an *proof table* is the number of its configurations.

A proof of a wff B is a closed proof table starting from the configuration $\{\mathbf{F}B\}$.

A finite set of swff's S is *consistent* iff no *proof table* starting from S is closed.

Let H be an swff. We call *extension(s)* of H the set(s) $\mathcal{R}^1_H, \dots, \mathcal{R}^n_H$ ($n \geq 1$) coinciding with the sets in the configuration obtained by applying the rule related to H in T to the configuration $\{H\}$, i.e., the extensions of the swff $H \equiv \mathbf{T}(A \to B)$ are the sets $\mathcal{R}^1_H = \{\mathbf{F}A, \mathbf{T_{cl}}B\}, \mathcal{R}^2_H = \{\mathbf{T}B\}$ and $\mathcal{R}^3_H = \{\mathbf{F_c}A\}$.

3.2 Soundness and Completeness results

It is easy to check that the rules of Table 1 preserve realizability, and hence, using Proposition 1, the Soundness Theorem can be proved in the usual way ([1]).

Theorem 2 (Soundness of T). *If a proof table starting from a swff* $\mathbf{F}A$ *is closed, then* $A \in \mathbf{HT}$.

The completeness proof is based (as usual) on a general method allowing to build up models for consistent sets of swff's. Given a consistent set of swff's S, we will construct a model $\underline{K}_{\mathbf{HT}}(S)$ on whose root the set S will be realized.

Let $A_1, \dots, A_n$ be any listing of swff's of S (without repetitions of swff's). Starting from this enumeration we construct the following sequence $\{S_i\}_{i\in\omega}$ of sets of swff's.

- $S_0 = S$;
- Let $S_i = \{H_1, \dots, H_u\}$; then

$$S_{i+1} = \bigcup_{H_j \in S_i} \mathcal{U}(H_j, i),$$

where, setting

$$S_j' = \mathcal{U}(H_1, i) \bigcup \cdots \bigcup \mathcal{U}(H_{j-1}, i) \bigcup \{H_j, \ldots, H_u\}$$

where: $\mathcal{U}(H_j, i)$ is a extension $\mathcal{R}_{H_j}$ of H_j such that $\mathcal{U}(H_1, i) \cup \cdots \cup \mathcal{U}(H_{j-1}, i) \cup \{H_{j+1}, \ldots, H_k, A_{i+1}, A_{i+2}, \ldots\} \cup \mathcal{R}_{H_j}$ is consistent.

Now, by induction on $i \geq 0$, it is easy to prove that if S is consistent then any S_i is consistent. Moreover, since S is finite there exists an index j such that $S_i = S_j$ for any $i \geq j$. Let u be the first index such that $S_u = S_{u+1}$. We call S_u the *node set of* S and we denote it with $\overline{S}$. Moreover we call $\{S_0, \ldots, S_u\}$ the *sequence generating* $\overline{S}$.

Now, we define

$$\begin{aligned} \Gamma_0 &= \{H \in \overline{S} \mid H \equiv \mathbf{T}A\} \\ \Gamma_1 &= \{H \in \overline{S} \mid H \equiv \mathbf{T}A \text{ or } H \equiv \mathbf{T_{cl}}A\} \end{aligned}$$

Given a consistent (and finite) set of swff's S, we define a (finite) structure $\underline{K}(S) = \langle P, \leq, \Vdash \rangle$ as follows:

1. $P = \{\Gamma_0, \Gamma_1\}$;
2. $\Gamma_0 \leq \Gamma_1$ and $\Gamma_i \leq \Gamma_i$ for $i \in \{0, 1\}$;
3. For any $\alpha \in P$ and for any propositional variable p, $\alpha \Vdash p$ iff $\mathbf{T}p \in \alpha$ or $\mathbf{T_{cl}}p \in \alpha$.

We remark that, given a consistent set of swff's S, in general we can build different saturated sets for S, and hence we can build different models $\underline{K}(S)$; however, all these structures are equivalent with respect to our purpose. It is obvious that $\underline{K}(S)$ is a model.

Now, we prove the Fundamental Lemma for **HT**.

Lemma 1. *Let S be a consistent set of swff's and let $\underline{K}(S) = \langle P, \leq, \Vdash \rangle$ be defined as above. For any swff $H \in S_i$ $(0 \leq i \leq u)$, $\Gamma_0 \rhd H$ in $\underline{K}(S)$.*

Proof. The proof goes by induction on the degree of the swff H (Definition 2).

Basis: For $\deg(H) = 0$ we have that $H \equiv \mathcal{S}p$ with p a propositional variable and, by construction, $H \in S_u$. Now, if $\mathcal{S} \equiv \mathbf{T}$, $\mathbf{T}p \in S_i$ then, by definition of $\underline{K}(S)$, $\Gamma_0 \rhd H$. If $\mathcal{S} \equiv \mathbf{F}$, since S_u is consistent, $\mathbf{T}p \notin S_u$ and hence $\Gamma_0 \rhd \mathbf{F}p$ by our definition of forcing. If $\mathcal{S} \equiv \mathbf{F_c}$, since S_u is consistent neither $\mathbf{T}p$ nor $\mathbf{T_{cl}}p$ belong to S_u, hence $\Gamma_0 \nVdash p$ and $\Gamma_1 \nVdash p$; this implies

$\Gamma_0 \rhd \mathbf{F_c}p$. Finally, if $\mathcal{S} \equiv \mathbf{T_{cl}}$, then $\mathbf{T_{cl}}p \in \Gamma_1$ and, by definition of the forcing relation, we have that $\Gamma_1 \Vdash p$, that is $\Gamma_0 \rhd \mathbf{T_{cl}}p$.

Step: Now, let us assume that the assertion holds for any swff $H' \in S_i$ with degree less than or equal to h and let us suppose that $\deg(H) = h+1$. The proof goes by cases, according to the form of the swff H. Here we give only some illustrative examples.

Case $H \equiv \mathbf{T}(A \wedge B)$: $\mathbf{T}(A \wedge B) \in S_i$ implies that there exists $S_j \in \{S_{i+1} \dots S_u\}$ such that $\{\mathbf{T}A, \mathbf{T}B\} \subseteq S_j$. Thus, we get, by the induction hypothesis, that $\Gamma_0 \Vdash A$ and $\Gamma_0 \Vdash B$, hence $\Gamma_0 \Vdash A \wedge B$ which means $\Gamma_0 \rhd \mathbf{T}(A \wedge B)$.

Case $H \equiv \mathbf{F}(A \rightarrow B)$: $\mathbf{F}(A \rightarrow B) \in S_i$ implies that there exists $S_j \in \{S_{i+1} \dots S_u\}$ such that either $\{\mathbf{T}A, \mathbf{F}B\} \subseteq S_j$ or $\{\mathbf{T_{cl}}A, \mathbf{F_c}B\} \subseteq S_j$. In the first case, we get, by the induction hypothesis, that $\Gamma_0 \Vdash A$ and $\Gamma_0 \nVdash B$, hence $\Gamma_0 \nVdash A \rightarrow B$. In the latter case, we get, by induction hypothesis, $\Gamma_1 \Vdash A$ and $\Gamma_1 \nVdash B$, therefore $\Gamma_0 \nVdash A \rightarrow B$ which means $\Gamma_0 \rhd \mathbf{F}(A \rightarrow B)$.

Case $H \equiv \mathbf{T_{cl}}(A \wedge B)$: $\mathbf{T_{cl}}(A \wedge B) \in S_i$ implies that there exists $S_j \in \{S_{i+1} \dots S_u\}$ such that $\{\mathbf{T_{cl}}A, \mathbf{T_{cl}}B\} \subseteq S_j$. Thus, we get, by the induction hypothesis, that $\Gamma_1 \Vdash A$ and $\Gamma_1 \Vdash B$, hence $\Gamma_1 \Vdash A \wedge B$ which means $\Gamma_0 \Vdash \not(A \wedge B)$ therefore $\Gamma_0 \rhd \mathbf{T_{cl}}(A \wedge B)$.

Completeness of the calculus follows from the Lemma above.

Theorem 3 (Completeness of T). *If $A \in$ **HT**, then there exists a closed proof table starting from* $\mathbf{F}A$.

Proof: Suppose the assertion is not true. Then, $\{\mathbf{F}A\}$ is a *consistent* set of swff's. By Lemma 1 this implies that $\mathbf{F}A$ is *-realizable* and we get a contradiction. $\Box$

We remark that the function $\deg(H)$ is linearly bounded by the length of H. Moreover, for every swff H, $deg(\mathcal{R}^i_H) \leq \deg(H)$ which implies that the depth of every proof table is linearly bounded in the length of the formula to be proved.

3.3 Complexity of the Tableaux method

As it is well-known, implication in **HT** logic takes exponential time in general. More precisely, checking Strong Equivalence is a *coNP*-complete problem.

In the past, some of these authors have investigated the space complexity of decision procedures for the interpolable logics. In this context results quite in line with the ones of [8] have been obtained in [5,6], where proofs are given that the decision procedures for the interpolable propositional intermediate logics are $O(n \log n)$-SPACE.

4 Other approaches

Several alternative approaches to the study of SE have stemmed from [12] and the debate that took place on the TAG[3] mailing list. Two issues are mainly investigated today: alternative characterizations of SE and extension of the concept to nested programs, i.e., those where head and body of rules are allowed to be arbitrary formulae. Nested programs are not considered in this article. The alternative characterizations of Strong Equivalence for normal logic programs, however, are very enlightening and very technical. Several authors, e.g. [2,14,15] and maybe others, characterize SE in terms of equivalence of the programs w.r.t. a different logical system, such as Pertinence and Intuitionistic logic.

Turner [20] gives an alternative characterization of strong equivalence in terms of SE-models, which are pairs (X, Y) of consistent sets of literals where $X \subseteq Y$ and, for a program Π, both X and Y are closed under Π^Y, which is the standard Gelfond-Lifschitz reduct [9] of Π relative to Y. SE-models are shown to correspond to models in the logic of Here-and-There. However, Turner's is the only approach we are aware of which studies SE within logic programming proper, although it relies, differently from this present work, on a new model theory.

5 Conclusions

The tableau calculus for HT given in [1] and described above has been implemented by using ANSI C language. The source code can be downloaded at the following URL:
`http://enigma.dimequant.unimib.it/_HT/index.html`.

The prover has been implemented following the proof of the completeness theorem. In order to keep the number of nodes of the proof tables low, in our strategy first we apply the rules with one extension ($\mathbf{T}\wedge, \mathbf{T_{cl}}\wedge, \mathbf{T}\neg, \mathbf{T_{cl}}\neg, \mathbf{F}\vee, \mathbf{F_c}\vee, \mathbf{F_c} \rightarrow \mathbf{F}\neg, \mathbf{F_c}\neg$) and, if no rule with one extension can be applied, we apply the rules with two or more extensions ($\mathbf{T}\vee, \mathbf{T_{cl}}\vee, \mathbf{T}\rightarrow, \mathbf{T_{cl}}\rightarrow, \mathbf{F}\wedge, \mathbf{F_c}\wedge, \mathbf{F}\rightarrow$).

[3] Texas Actions group: *http://www.cs.utexas.edu/users/vl/tag/*

In [18] a tableau calculus for the logic here-and-there with strong negation is described, where strong negation is treated by introducing in the language a new connective. Although, as it is well known, by Gurevich's trick one can eliminate strong negation by doubling the predicates, in logical terms it is elegant and more complete to keep it in as a connective. The calculus of [18] without strong negation is equivalent to the one described in this paper and introduced in [1], and characterizes the 3-valued logic of Here-and-There. A straightforward translation between the two calculi can be obtained by associating the four signs $\mathbf{T}, \mathbf{F}, \mathbf{F_c}$ and $\mathbf{T_{cl}}$ to sets of numbers. For instance, the swff $\mathbf{F}A$, that represents non-truth of A, is represented in [18] by the pair $\{0, 1\}$ of truth values (standing for falsity and non-truth respectively). Similarly, the other signs are translated as follows: $\mathbf{T} = \{2\}$, $\mathbf{T_{cl}} = \{1, 2\}$ and $\mathbf{F_c} = \{0\}$.

Acknowledgments

We acknowledge useful discussions and much encouragement from Michael Gelfond, Mario Ornaghi and David Pearce.

References

1. A. Avellone, M. Ferrari, and P. Miglioli. Duplication-free tableau calculi and related cut-free sequent calculi for the interpolable propositional intermediate logics. *Logic Journal of the IGPL*, 7(4):447–480, 1999.
2. P. Cabalar, 2001. *Well-founded Semantics as Two-dimensional Here and There.* Proc. of AAAI Spring Symposium 2001 on Answer Set Programming (ASP2001), pp. 15–20. AAAI Press, Tech. report SS-01-01..
3. A. Chagrov and M. Zakharyaschev, 1997. *Modal Logic.* Oxford Univ. Press.
4. C. Fiorentini and P. Miglioli, 1999. A cut-free sequent calculus for the logic of constant domains. Tech. report, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano.
5. G. Fiorino. An $O(n \log n)$-SPACE decision procedure for the propositional Dummett Logic. *Journal of Automated Reasoning*, 27: 297-311, 2001.
6. G. Fiorino. Space-efficient Decision Procedures for Three Interpolable Propositional Intermediate Logics. *Journal of Logic and Computation*, Vol 12, Issue 6, 2002.
7. M.C. Fitting, 1969. *Intuitionistic Logic, Model Theory and Forcing.* North-Holland.
8. J. Hudelmaier, 1993. An $O(n \log n)$-SPACE decision procedure for intuitionistic propositional logic. *Journal of Logic and Computation*, 3(1):63–75.
9. Gelfond, M. and Lifschitz, V., 1988. *The stable model semantics for logic programming.* Proc. of 5th ILPS conference, MIT Press, pp. 1070–1080.
10. Gelfond, M. and Lifschitz, V., 1991. *Classical negation in logic programs and disjunctive databases.* New Generation Computing, pp. 365–387.

11. T. Janhunen, T. and E. Oikarinen, 2002. *Testing the Equivalence of Logic Programs under Stable Model Semantics.* In Proc. of JELIA 2002, 8th European Conf. on Logics in Artificial Intelligence, Springer-Verlag LNAI 2424, pp 493–504.
12. Lifschitz, V., Pearce, D. and Valverde, A., 2001. Strongly Equivalent logic programs. ACM Transactions on Computational Logic, 2:526–541.
13. Marek, W., and Truszczyński, M., 1999. *Stable models and an alternative logic programming paradigm,* In: The Logic Programming Paradigm: a 25-Year Perspective, pp. 375–398 Springer-Verlag.
14. Osorio, M., Navarro, J. A. and Arrazola, J., 2001. *Equivalence in Answer Set programming.* Annals of pure and Applied Logic, 108:1-3, pp. 153–188. Also in Proc. of LOPSTR 2001, pp. 18–28.
15. Otero, R., 2001. *Pertinence Logic Characterization of Stable Models.* Proc. of AAAI Spring Symposium 2001 on Answer Set Programming (ASP2001), pp. 153–159. AAAI Press, Tech. report SS-01-01..
16. Pearce, D., 1997. A New Logical Characterization of Stable Models and Answer Sets. *Non-Monotonic Extensions of Logic Programming*, pp. 55–70, Springer-Verlag LNAI 1216.
17. Pearce, D., 1999. *Stable Inference as Intuitionistic Validity.* J. of Logic Programming, 38, pp. 79–91.
18. Pearce, D., de Guzmán, I. P., Valverde, A. 2000. *A Tableau Calculus for Equilibrium Entailment.* In R. Dyckhoff, editor, *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 1847 of *LNCS*, pages 352–367. Springer-Verlag, 2000.
19. Web location of the most known ASP solvers.
CCALC: *http://www.cs.utexas.edu/users/mcain/cc*
Cmodels: *http://www.cs.utexas.edu/users/tag/cmodels.html*
DeReS: *http://www.cs.engr.uky.edu/~lpnmr/DeReS.html*
DLV: *http://www.dbai.tuwien.ac.at/proj/dlv/*
NoMoRe: *http://www.cs.uni-potsdam.de/ linke/nomore/*
smodels: *http://www.tcs.hut.fi/Software/smodels/*
20. Turner, H., 1997. Strong Equivalence for Logic Programs and Default Theories (Made Easy). T. Eiter and W. Faber (Eds.) *Logic Programming and NonMonotonic Reasoning.* Springer-Verlag LNAI 2173, pp. 55–70.
21. Van Gelder A., Ross K.A. and Schlipf J., 1990. *The Well-Founded Semantics for General Logic Programs.* Journal of the ACM 38:3.