Forgetting Auxiliary Atoms in Forks*

Felicidad Aguado[1], Pedro Cabalar[1], Jorge Fandinno[1,2], David Pearce[3], Gilberto Pérez[1] and Concepción Vidal[1]

[1] Department of Computer Science
University of Corunna, SPAIN
{aguado,cabalar,jorge.fandino}@udc.es
{gilberto.pvega,concepcion.vidalm}@udc.es

[2] Institut de recherche en informatique de Toulouse
Toulouse Universtiy, FRANCE
jorge.fandinno@irit.fr

[3] Universidad Politécnica de Madrid, SPAIN
david.pearce@upm.es

Abstract. In this work we tackle the problem of checking strong equivalence of logic programs that may contain local auxiliary atoms, to be removed from their stable models and to be forbidden in any external context. We call this property *projective strong equivalence* (PSE). It has been recently proved that not any logic program containing auxiliary atoms can be reformulated, under PSE, as another logic program or formula without them – this is known as *strongly persistent forgetting*. In this paper, we introduce a conservative extension of *Equilibrium Logic* and its monotonic basis, the logic of *Here-and-There*, in which we deal with a new connective '|' we call *fork*. We provide a semantic characterisation of PSE for forks and use it to show that, in this extension, it is always possible to forget auxiliary atoms under strong persistence. We further define when the obtained fork is representable as a regular formula.

1 Introduction

Answer Set Programming (ASP [1]) has become an established problem-solving paradigm for Knowledge Representation and Reasoning (KRR). The reasons for this success derive from the practical point of view, with the availability of efficient solvers [2, 3] and application domains [4], but also from its solid theoretical foundations, rooted in the *stable models* [5] semantics for normal logic programs that was later generalised to arbitrary propositional [6], first-order [7,

* This work has been partially supported by Xunta de Galicia, Spain (projects GPC ED431B 2016/035 and 2016-2019 ED431G/01 for CITIC center) and European Regional Development Fund (ERDF); and by Centre International de Mathématiques et d'Informatique de Toulouse (CIMI) through contract ANR-11-LABEX-0040-CIMI within the program ANR-11-IDEX-0002-02

8] and infinitary [9] formulas. An important breakthrough that supported these extensions of ASP has been its logical characterisation in terms of *Equilibrium Logic* [6] and its monotonic basis, the intermediate logic of *Here-and-There* (HT). Despite its expressiveness, a recent result [10] has shown that Equilibrium Logic has limitations in capturing the representational power of auxiliary atoms, which cannot always be forgotten. To illustrate this point, take the following problem.

Example 1. Two individuals, mother and father, both carrying alleles a and b, procreate an offspring. We want to generate all the possible ways in which the offspring may inherit its parents' genetic information.

According to Mendelian laws, we should obtain three possible combinations that, ignoring their frequency, correspond to the sets of alleles $\{a\}$, $\{b\}$ and $\{a, b\}$. The straightforward way to generate these three stable models is to encode an inclusive disjunction with the following 3 rules:

$$a \vee \neg a \qquad b \vee \neg b \qquad \bot \leftarrow \neg a \wedge \neg b \tag{P_1}$$

A drawback of this representation is that it does not differentiate the information coming from each parent, possibly becoming a problem of elaboration tolerance. For instance, if only the mother's information were available, one would expect to obtain the stable models $\{a\}$ and $\{b\}$ but not $\{a, b\}$, as there is no evidence of that combination without further information about the father. So, the mother alone would be better represented by a regular disjunction $a \vee b$. However, we cannot represent each parent as an independent disjunction like that, since $(a \vee b) \wedge (a \vee b)$ just amounts to $(a \vee b)$ and the combination $\{a, b\}$ is not obtained. A simple way to represent these two disjunctions separately is using auxiliary atoms to keep track of alleles from the mother ($ma \vee mb$) and the father ($fa \vee fb$). This leads to program P_2:

$$ma \vee mb \qquad a \leftarrow ma \qquad b \leftarrow mb \tag{P_m}$$
$$fa \vee fb \qquad a \leftarrow fa \qquad b \leftarrow fb \tag{P_f}$$

consisting of the mother's contribution P_m and the father's contribution P_f. Four stable models are obtained from P_2, $\{ma, fa, a\}$, $\{mb, fb, b\}$, $\{ma, fb, a, b\}$ and $\{mb, fa, a, b\}$, but if we *project* them on the original vocabulary $V = \{a, b\}$ (i.e. we remove auxiliary atoms), they collapse to three $\{a\}$, $\{b\}$ and $\{a, b\}$ as expected. Note that, although auxiliary atoms in this example have a meaning in the real world (they represent the effective sources of each inherited allele) they were not part of the original alphabet $V = \{a, b\}$ of Example 1, which does not distinguish between the same effect $\{a, b\}$ but due to different sources $\{ma, fb, a, b\}$ and $\{mb, fa, a, b\}$.

As we have seen, P_1 and P_2 are "V-equivalent" in the sense that they yield the same stable models when projected to alphabet $V = \{a, b\}$. A natural question is whether this also holds in any context, that is, if $P_1 \cup Q$ and $P_2 \cup Q$ also yield the same V-projected stable models, for any context Q in the target alphabet V (since we want to keep auxiliary atoms local or hidden). This is obviously a kind of *strong equivalence* relation [11] – in fact it is one of the possible generalisations[1] of strong equivalence studied in [12]. In this paper, we will just call it *projective strong equivalence* (PSE) with respect to V, or V*-strong equivalence* for short. The PSE relation has also been used in the literature for comparing a program P and some transformation $tr(P)$ that either extends the vocabulary with new auxiliary atoms [13] (called there *strong faithfulness*) or reduces it for forgetting atoms as in [10] (called there *strong persistence*).

As we will see later, programs P_1 and P_2 are indeed V-strongly equivalent, so they express the same combined knowledge obtained from both parents. However, if we want to keep program P_m alone capturing the mother's contribution, *there is no possible* $\{a, b\}$*-strongly equivalent representation* in Equilibrium Logic (the same happens with P_f). In other words, we cannot forget atoms ma and mb in P_m and get a program preserving PSE. This impossibility follows from a recent result in [10] that shows that forgetting atoms under strong persistence is sometimes impossible. In practice, this means that auxiliary atoms in ASP are more than 'just' auxiliary, as they allow one to represent problems that cannot be captured without them. A natural idea is to consider an extension of ASP in which forgetting auxiliary atoms is always possible.

In this paper, we extend logic programs to include a new construct '$\mid$' we call *fork* whose intuitive meaning is that the stable models of $P \mid P'$ correspond to the union of stable models from P and P' in any context[2] Q, that is $SM[(P \mid P') \wedge Q] = SM[P \wedge Q] \cup SM[P' \wedge Q]$. Using this construct, we can represent Example 1 as the conjunction of two forks $(a \mid b) \wedge (a \mid b)$, one per each parent. This conjunction of forks is not idempotent but will actually amount to $(a \mid b \mid a \wedge b)$. We will show that forgetting is always possible in forks but some of them, such as $(a \mid b)$, cannot be represented in Equilibrium Logic unless we allow the introduction of auxiliary atoms.

The rest of the paper is organised as follows. Section 2 recalls basic definitions of HT and Equilibrium Logic. Section 3 introduces an alternative characterisation of HT in terms of T*-supports.* The next section extends the syntax with the fork connective and generalises the semantics to sets of T-supports (so-called T*-views*). In Section 5 we characterise PSE for forks and relate this property to forgetting. Finally, Section 6 discusses related work and concludes the paper.

2 Preliminaries

We begin by recalling some basic definitions and results related to HT. Let At be a set of atoms called the *propositional signature.* A *(propositional) formula* φ is defined using the grammar:

$$\varphi \ ::= \ \bot \ \big| \ p \ \big| \ \varphi \wedge \varphi \ \big| \ \varphi \vee \varphi \ \big| \ \varphi \rightarrow \varphi$$

[1] It corresponds to *relativised* strong equivalence (with respect to V) with projection (with respect to V).

[2] For simplicity, we understand programs as the conjunction of their rules.

where p is an atom. We will use Greek letters φ, ψ, γ and their variants to stand for formulas. We define the derived operators $\neg\varphi \stackrel{\text{def}}{=} (\varphi \to \bot)$ and $\top \stackrel{\text{def}}{=} \neg\bot$ and $\varphi \leftrightarrow \psi \stackrel{\text{def}}{=} (\varphi \to \psi) \wedge (\psi \to \varphi)$. Given a formula φ, by $At(\varphi) \subseteq At$ we denote the set of atoms occurring in φ. For simplicity, we consider finite theories understood as the conjunction of their formulas. The extension to infinite theories is straightforward.

A *classical interpretation* T is a set of atoms $T \subseteq At$. We write $T \models \varphi$ to stand for the usual classical satisfaction of a formula φ. An HT-*interpretation* is a pair $\langle H, T\rangle$ (respectively called "here" and "there") of sets of atoms $H \subseteq T \subseteq At$; it is said to be *total* when $H = T$. The fact that an interpretation $\langle H, T\rangle$ *satisfies* a formula φ, written $\langle H, T\rangle \models \varphi$, is recursively defined as follows:

- $\langle H, T\rangle \not\models \bot$
- $\langle H, T\rangle \models p$ iff $p \in H$
- $\langle H, T\rangle \models \varphi \wedge \psi$ iff $\langle H, T\rangle \models \varphi$ and $\langle H, T\rangle \models \psi$
- $\langle H, T\rangle \models \varphi \vee \psi$ iff $\langle H, T\rangle \models \varphi$ or $\langle H, T\rangle \models \psi$
- $\langle H, T\rangle \models \varphi \to \psi$ iff both (i) $T \models \varphi \to \psi$ and (ii) $\langle H, T\rangle \not\models \varphi$ or $\langle H, T\rangle \models \psi$

By abuse of notation, we use '$\models$' both for classical and for HT-satisfaction: the ambiguity is removed by the form of the left interpretation (a single set T for classical and a pair $\langle H, T\rangle$ for HT). It is not difficult to see that, for total interpretations, $\langle T, T\rangle \models \varphi$ amounts to classical satisfaction $T \models \varphi$.

Proposition 1. *Any HT-interpretation $\langle H, T\rangle$ and any formula φ satisfy that $\langle H, T\rangle \models \varphi$ implies $\langle T, T\rangle \models \varphi$ (i.e. $T \models \varphi$ classically).*

HT-models can also be computed using Ferraris' reduct [14], φ^T, defined as the result of replacing by $\bot$ those maximal subformulas of φ that are not (classically) satisfied by interpretation T. As an example, given $\varphi = (\neg a \to b)$ we have the reducts $\varphi^{\emptyset} = \bot$, $\varphi^{\{a\}} = (\bot \to \bot)$, $\varphi^{\{b\}} = (\neg\bot \to b)$ and $\varphi^{\{a,b\}} = (\bot \to b)$. The correspondence with HT-satisfaction is given by:

Proposition 2 (Lemma 1, [14]). *Given $H \subseteq T$: $\langle H, T\rangle \models \varphi$ iff $H \models \varphi^T$.*

A total interpretation $\langle T, T\rangle$ is an *equilibrium model* of a formula φ iff $\langle T, T\rangle \models \varphi$ and there is no $H \subset T$ such that $\langle H, T\rangle \models \varphi$. If so, we say that T is a *stable model* of φ. By Proposition 2 this means that T is a stable model of φ iff it is a minimal classical model of φ^T. We write $SM[\varphi]$ to stand for the set of stable models of φ. Moreover, we represent their projection onto some vocabulary V as $SM_V[\varphi] \stackrel{\text{def}}{=} \{\, T \cap V \mid T \in SM[\varphi]\,\}$.

Definition 1 (projective strong entailment/equivalence). *Let φ and ψ be formulas and V some vocabulary (set of atoms). We say that φ V*-strongly entails *ψ, written $\varphi \mathrel{|\!\!\sim_V} \psi$ if $SM_V[\varphi \wedge \gamma] \subseteq SM_V[\psi \wedge \gamma]$ for any formula γ such that $At(\gamma) \subseteq V$. We further say that φ and ψ are V*-strongly equivalent*, written $\varphi \cong_V \psi$, if both $\varphi \mathrel{|\!\!\sim_V} \psi$ and $\psi \mathrel{|\!\!\sim_V} \varphi$, that is, $SM_V[\varphi \wedge \gamma] = SM_V[\psi \wedge \gamma]$ for any formula γ such that $At(\gamma) \subseteq V$.*

When the vocabulary $V \supseteq At(\varphi) \cup At(\psi)$ contains the original language of φ and ψ, the projection has no relevant effect and the previous definitions amount to regular (non-projective) *strong entailment* and *strong equivalence*. In this case, we simply drop the V subindex in the previous notations. The following results, respectively proved in [11] and [15], characterise non-projective strong equivalence and entailment in terms of HT:

Proposition 3 (From [11] and [15]). *Let φ, ψ be a pair of formulas. Then*
(i) *$\varphi \cong \psi$ iff φ and ψ are equivalent in* HT.
(ii) *$\varphi \mid\!\sim \psi$ iff both φ classically entails ψ and, for any H, if $\langle H, T\rangle \models \psi$ and $T \models \varphi$ then $\langle H, T\rangle \models \varphi$.*

In the case of projected strong entailment and equivalence, a semantic characterisation was provided in [12], although limited to the case of disjunctive logic programs. We will provide later a characterisation of strong entailment and equivalence for fork formulas that, for the particular case in which the fork operator does not occur, will also constitute an extension of [12] to arbitrary propositional formulas.

3 T-supports

In this section we consider an alternative characterisation of HT-semantics that relies on the idea of *support* for a given classical interpretation T.

Definition 2 (T-support). *Given a set T of atoms, a T-*support *$\mathcal{H}$ is a set of subsets of T, that is $\mathcal{H} \subseteq 2^T$, satisfying $T \in \mathcal{H}$ if $\mathcal{H} \neq \emptyset$.*

To increase readability of examples, we will just write a support as a sequence of interpretations between square brackets. For instance, possible supports for $T = \{a, b\}$ are $[\{a, b\}\ \{a\}]$, $[\{a, b\}\ \{b\}\ \emptyset]$ or the empty support $[\ ]$.

Intuitively, $\mathcal{H}$ will be used to capture the set of "here" components H that *support* the "there" world T as a model of a given formula φ, that is, the set of H's such that $\langle H, T\rangle \models \varphi$. When $\mathcal{H}$ is empty $[\ ]$, there is no support for T, so $\langle T, T\rangle \not\models \varphi$ and thus, T is not even a classical model. If $\mathcal{H}$ is not empty, this means we have at least some model $\langle H, T\rangle$ and, by Proposition 1, $\langle T, T\rangle$ must be a model too; this is why we require $T \in \mathcal{H}$ in the set. When not empty, the fewer models in $\mathcal{H}$, the more supported is T, since it is closer to being stable. Accordingly, the most supported $\mathcal{H}$ would precisely be $\mathcal{H} = [\ T\ ]$ corresponding to a stable model. This ordering relation is formally defined below.

Definition 3. *Given a set $T \subseteq At$ of atoms and two T-supports $\mathcal{H}$ and $\mathcal{H}'$ we write $\mathcal{H} \preceq_T \mathcal{H}'$ iff either $\mathcal{H} = [\ ]$ or $\mathcal{H} \supseteq \mathcal{H}' \neq [\ ]$.*

We usually write $\mathcal{H} \preceq \mathcal{H}'$ instead of $\mathcal{H} \preceq_T \mathcal{H}'$ when clear from the context. As an example, the classical interpretation $T = \{a, b\}$ is more supported in $\mathcal{H}_1 = [\{a, b\}\ \{a\}]$ than in $\mathcal{H}_2 = [\{a, b\}\ \{a\}\ \{b\}\ \emptyset]$, that is $\mathcal{H}_2 \preceq \mathcal{H}_1$, because $\mathcal{H}_2$ contains additional interpretations and is further from being stable. We write

$\mathbf{H}_T$ to stand for the set of all possible T-supports. $(\mathbf{H}_T, \preceq)$ forms a poset whose bottom and top elements are $[\,]$ and $[\ T\ ]$ respectively. Given a T-support $\mathcal{H}$, we define its complementary support $\overline{\mathcal{H}}$ as:

$$\overline{\mathcal{H}} \stackrel{\text{def}}{=} \begin{cases} [\,] & \text{if } \mathcal{H} = 2^T \\ [\ T\ ] \cup \{H \subseteq T \mid H \notin \mathcal{H}\} & \text{otherwise} \end{cases}$$

We also define $\mathcal{H}_V$ as the projection of every set in $\mathcal{H}$ to the vocabulary V, i.e., $\mathcal{H}_V \stackrel{\text{def}}{=} \{H \cap V \mid H \in \mathcal{H}\}$. The relation between T-supports and formulas is given by the following definition.

Definition 4 (T-denotation). *Let $T \subseteq At$. The* T-denotation *of a formula φ, written $[\![\, \varphi \,]\!]^T$, is a T-support recursively defined as follows:*

$$[\![\, \bot \,]\!]^T \stackrel{\text{def}}{=} [\,] \qquad\qquad [\![\, \varphi \wedge \psi \,]\!]^T \stackrel{\text{def}}{=} [\![\, \varphi \,]\!]^T \cap [\![\, \psi \,]\!]^T$$

$$[\![\, p \,]\!]^T \stackrel{\text{def}}{=} \{H \subseteq T \mid p \in H\} \qquad\qquad [\![\, \varphi \vee \psi \,]\!]^T \stackrel{\text{def}}{=} [\![\, \varphi \,]\!]^T \cup [\![\, \psi \,]\!]^T$$

$$[\![\, \varphi \to \psi \,]\!]^T \stackrel{\text{def}}{=} \begin{cases} [\,] & \text{if } T \not\models \varphi \to \psi \\ \overline{[\![\, \varphi \,]\!]^T} \cup [\![\, \psi \,]\!]^T & \text{otherwise} \end{cases}$$

The following proposition follows by structural induction and shows that T-denotations can be used as an alternative semantics for the logic HT.

Proposition 4. *For any interpretation $\langle H, T\rangle$ and formula φ:*
$\langle H, T\rangle \models \varphi$ iff $H \in [\![\, \varphi \,]\!]^T$.

Corollary 1. *For any set T of atoms and propositional formulas φ, ψ, the following conditions hold:*
(i) *$T \models \varphi$ iff $[\![\, \varphi \,]\!]^T \neq [\,]$ iff $T \in [\![\, \varphi \,]\!]^T$.*
(ii) *T is a stable model of φ iff $[\![\, \varphi \,]\!]^T = [\ T\ ]$.*
(iii) *Given $H \subseteq T$: $H \in [\![\, \varphi \,]\!]^T$ iff $H \models \varphi^T$.*

The last item asserts that the denotation $[\![\, \varphi \,]\!]^T$ can also be seen as a semantic counterpart of Ferraris' reduct φ^T. The following result is a rephrasing of Theorem 2 in [16] under the equivalence of Proposition 4 and asserts that any assignment of T-supports for all T's corresponds to (the denotation of) some formula.

Proposition 5. *Let σ be an arbitrary assignment of a support $\sigma(T) \in \mathbf{H}_T$ for each $T \subseteq At$. Then, there exists a formula φ s.t. $[\![\, \varphi \,]\!]^T = \sigma(T)$ for all T.*

As an example, given $At = \{a, b\}$, the T-denotations of the formulas $\neg a \to b$ and $a \vee b$ are:

T	$[\![\, \neg a \to b \,]\!]^T$	$[\![\, a \vee b \,]\!]^T$
$\emptyset$	$[\,]$	$[\,]$
$\{a\}$	$[\ \emptyset\ \{a\}\ ]$	$[\ \{a\}\ ]$
$\{b\}$	$[\ \{b\}\ ]$	$[\ \{b\}\ ]$
$\{a, b\}$	$[\{a, b\}\ \{a\}\ \{b\}\ \emptyset\ ]$	$[\ \{a, b\}\ \{a\}\ \{b\}\ ]$

Notice that the only stable model of $\neg a \rightarrow b$ is $\{b\}$ because $[\![\neg a \rightarrow b]\!]^{\{b\}} = [\{b\}]$. Similarly, the stable models of $a \vee b$ are $\{a\}$ and $\{b\}$. Note also that, for all T, we always have $[\![\neg a \rightarrow b]\!]^T \preceq [\![a \vee b]\!]^T$, that is, $\neg a \rightarrow b$ is always less supported (further from being stable) than $a \vee b$. In fact, this has an interesting consequence, as stated by the next result:

Proposition 6. *For any two propositional formulas φ, ψ the following hold:*
(i) *$\varphi \mathrel{|\!\!\sim} \psi$ iff $[\![\varphi]\!]^T \preceq [\![\psi]\!]^T$ for every set $T \subseteq At$ of atoms,*
(ii) *$\varphi \cong \psi$ iff $[\![\varphi]\!]^T = [\![\psi]\!]^T$ for every set $T \subseteq At$ of atoms.*

While (ii) is an immediate consequence of Proposition 4, item (i) states that φ strongly entails ψ iff the former is always less supported than the latter. Note how Proposition 6 is much more readable than Proposition 3, especially regarding strong entailment and its relation to strong equivalence. In our example above, Proposition 6 implies that if we replace $\neg a \rightarrow b$ by $a \vee b$ in any program, we will get the same or perhaps more stable models.

4 Forks and *T*-views

A *fork* is defined using the grammar:

$$F \ ::= \ \bot \ \Big| \ p \ \Big| \ F \mid F \ \Big| \ F \wedge F \ \Big| \ \varphi \vee \varphi \ \Big| \ \varphi \rightarrow F$$

where φ is a propositional formula and $p \in At$ is an atom. As we can see, the fork operator '$|$' cannot occur in the scope of disjunction or negation, since $\neg F$ stands for $F \rightarrow \bot$ and implications do not allow forks in the antecedent. Extending the semantics to arbitrarily nested connectives is left for future work.

As we will see, a fork F will be always reducible to the form $(\varphi_1 | \dots | \varphi_n)$ where each φ_i is a formula. Thus, a natural way to define its semantics is keeping a set of supports $\Delta = \{\mathcal{H}_1, \dots, \mathcal{H}_n\}$ for each classical interpretation T. However, if we have a pair of supports in Δ such that $\mathcal{H}_i \preceq \mathcal{H}_j$, then $\mathcal{H}_i$ is useless since, according to Proposition 6 ii), if it yields a stable model, the latter is always produced by $\mathcal{H}_j$ too. For this reason, we will collect sets of supports that are $\preceq$-closed, so their *maximal* elements become the representative ones. Formally, given a T-support $\mathcal{H}$ we define the set of $\preceq$-smaller supports $\downarrow\!\mathcal{H} = \{\mathcal{H}' \in \mathbf{H}_T \mid \mathcal{H}' \preceq \mathcal{H}\}$. This is usually called the *ideal* of $\mathcal{H}$. We extend this notation to any set of supports Δ so that ${}^{\downarrow}\Delta \overset{\text{def}}{=} \bigcup_{\mathcal{H} \in \Delta} \downarrow\!\mathcal{H} = \{ \mathcal{H}' \preceq \mathcal{H} \mid \mathcal{H} \in \Delta \}$.

Definition 5 (*T*-view). *A T-*view *is a non-empty set of T-supports $\Delta \subseteq \mathbf{H}_T$ that is $\preceq$-closed, i.e., ${}^{\downarrow}\Delta = \Delta$.*

Notice that, since a T-view Δ is not empty and $\preceq$-closed, the smallest T-support $[\ ]$ is always included in Δ. Therefore, $\{[\ ]\}$ is the smallest T-view with respect to set inclusion. Analogously, if the greatest T-support $[\ T\ ]$ is included in Δ, then Δ is precisely $\downarrow\![\ T\ ]$. We are now ready to extend the concept of T-denotation to forks.

Definition 6 (T-denotation of a fork). *Let At be a propositional signature and $T \subseteq At$ a set of atoms. The T-*denotation *of a fork F, written $\langle\!\langle\, F \,\rangle\!\rangle^T$, is a T-view recursively defined as follows:*

$$
\begin{array}{rcl}
\langle\!\langle\, \bot \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & \{[\ ]\} \\
\langle\!\langle\, p \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & {\downarrow}[\![\, p \,]\!]^T \quad \textit{for any atom } p \\
\langle\!\langle\, F \wedge G \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & {}^{\downarrow}\!\{\ \mathcal{H} \cap \mathcal{H}' \ \big|\ \mathcal{H} \in \langle\!\langle\, F \,\rangle\!\rangle^T \ \textit{and}\ \mathcal{H}' \in \langle\!\langle\, G \,\rangle\!\rangle^T\ \} \\
\langle\!\langle\, \varphi \vee \psi \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & {}^{\downarrow}\!\{\ \mathcal{H} \cup \mathcal{H}' \ \big|\ \mathcal{H} \in \langle\!\langle\, \varphi \,\rangle\!\rangle^T \ \textit{and}\ \mathcal{H}' \in \langle\!\langle\, \psi \,\rangle\!\rangle^T\ \} \\
\langle\!\langle\, \varphi \to F \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & \begin{cases} \{[\ ]\} & \textit{if } T \models \varphi \textit{ and } \langle\!\langle\, F \,\rangle\!\rangle^T = \{[\ ]\} \\ {}^{\downarrow}\!\{\ \overline{[\![\, \varphi \,]\!]^T} \cup \mathcal{H} \ \big|\ \mathcal{H} \in \langle\!\langle\, F \,\rangle\!\rangle^T\ \} & \textit{otherwise} \end{cases} \\
\langle\!\langle\, F \mid G \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & \langle\!\langle\, F \,\rangle\!\rangle^T \cup \langle\!\langle\, G \,\rangle\!\rangle^T
\end{array}
$$

It is easy to see that the fork operator '$|$' is commutative, associative and idempotent. We will also see later that conjunction and implication distribute over '$|$'. As for the rest of operators, note that the definitions above also cover propositional formulas. The following result shows that this new T-denotation of a propositional formula φ as a T-view, $\langle\!\langle\, \varphi \,\rangle\!\rangle^T$, is precisely the ideal of its T-denotation as a T-support $[\![\, \varphi \,]\!]^T$.

Proposition 7. *Let φ be a propositional formula and $T \subseteq At$ be a set of atoms. Then, $\langle\!\langle\, \varphi \,\rangle\!\rangle^T = {\downarrow}[\![\, \varphi \,]\!]^T$.*

Corollary 2. *Given a propositional formula φ, a set $T \subseteq At$ of atoms is a stable model of φ iff $\langle\!\langle\, \varphi \,\rangle\!\rangle^T = {\downarrow}[\ T\ ]$.*

Corollary 2 provides a natural way to define stable models of forks.

Definition 7. *Given a fork F, we say that $T \subseteq At$ is a* stable model *of F iff $\langle\!\langle\, F \,\rangle\!\rangle^T = {\downarrow}[\ T\ ]$. $SM[F]$ denotes the set of stable models of F.*

The intuition behind a fork is that we can collect its stable models independently:

Proposition 8. *Given forks F and G: $SM[F \mid G] = SM[F] \cup SM[G]$.*

Once $SM[F]$ is defined, we can immediately extend the definition of V-strong entailment and equivalence to forks in the obvious way, i.e., using forks instead of propositional formulas. We postpone the effect of projecting onto some vocabulary V to the next section and focus on the regular, non-projected versions $\mid\!\sim$ and $\cong$. As in Proposition 6, $\mid\!\sim$ and $\cong$ have a simple characterisation in terms of denotations:

Proposition 9. *For any pair of forks F, G the following hold:*
(i) *$F \mid\!\sim G$ iff $\langle\!\langle\, F \,\rangle\!\rangle^T \subseteq \langle\!\langle\, G \,\rangle\!\rangle^T$ for every set $T \subseteq At$,*
(ii) *$F \cong G$ iff $\langle\!\langle\, F \,\rangle\!\rangle^T = \langle\!\langle\, G \,\rangle\!\rangle^T$ for every set $T \subseteq At$.*

This helps us to derive the following interesting properties:

Proposition 10. *Let F, G, H be arbitrary forks and φ a formula. Then:*

$$(F \mid G) \wedge H \cong (F \wedge H) \mid (G \wedge H) \tag{1}$$
$$\varphi \to (F \mid G) \cong (\varphi \to F) \mid (\varphi \to G) \tag{2}$$
$$(F \mid G) \cong G \qquad \text{if } F \mathrel{|\!\!\sim} G \tag{3}$$

The first two properties (1) and (2), allow us to reduce any arbitrary fork to a *normal form* $(\varphi_1 \mid \ldots \mid \varphi_n)$ where each φ_i is a propositional formula. Moreover, (3) is a kind of *subsumption* property that, without loss of generality, allows us to assume that no pair of formulas $\varphi_i \mathrel{|\!\!\sim} \varphi_j$ for $i \neq j$ in that expression. As an example of subsumption, take the fork $(\neg a \to b \mid a \vee b)$. As we saw before, $(\neg a \to b) \mathrel{|\!\!\sim} (a \vee b)$ because $[\![\neg a \to b]\!]^T \preceq [\![a \vee b]\!]^T$ for all T (Proposition 6). Then, the ideal $\downarrow\![\![\neg a \to b]\!]^T$ is included in $\downarrow\![\![a \vee b]\!]^T$ which (by Proposition 7) is the same as saying $\langle\!\langle \neg a \to b \rangle\!\rangle^T \subseteq \langle\!\langle a \vee b \rangle\!\rangle^T$. But then, $\langle\!\langle \neg a \to b \mid a \vee b \rangle\!\rangle^T = \langle\!\langle \neg a \to b \rangle\!\rangle^T \cup \langle\!\langle a \vee b \rangle\!\rangle^T = \langle\!\langle a \vee b \rangle\!\rangle^T$. In other words, $(\neg a \to b \mid a \vee b) \cong (a \vee b)$.

An important consequence of conjunction-distributivity (1) is that, although $\wedge$ is idempotent on propositional formulas, it ceases to be so when connecting forks. Take, for instance, the formalisation of Example 1 using the expression $(a \mid b) \wedge (a \mid b)$. If we apply distributivity and reduce to a normal form:

$$\begin{array}{rcll}
(a \mid b) \wedge (a \mid b) & \cong & a \wedge (a \mid b) \mid b \wedge (a \mid b) & \text{distributivity (1)} \\
& \cong & (a \mid a \wedge b) \mid (b \wedge a \mid b) & \text{distributivity (1), } \wedge\text{-idempotence} \\
& \cong & a \mid a \wedge b \mid b \wedge a \mid b & \text{associativity of `}|\text{'} \\
& \cong & a \mid a \wedge b \mid b & \text{commut. of } \wedge \text{ and idempotence of `}|\text{'}
\end{array}$$

but then, by Proposition 8, from $a \wedge b$ we get the stable model $\{a, b\} \notin SM[a \mid b]$.

5 Projective strong equivalence/entailment

In this section, we provide a semantic characterisation of projective strong entailment $\mathrel{|\!\!\sim}_V$ and equivalence $\cong_V$ for some vocabulary $V \subseteq At$. We say that a T-support $\mathcal{H}$ is[3] *V-vacuous* iff there is some $H \subset T$ in $\mathcal{H}$ satisfying $H \cap V = T \cap V$. The reason for the name "vacuous" is that if we take a formula φ with denotation $[\![\varphi]\!]^T = \mathcal{H}$, then T will never become stable if we can only add a context γ for vocabulary V. To do so, we would need $[\![\varphi \wedge \gamma]\!]^T = [\, T \,]$ but $H \subset T$ should also belong to the support since H and T are indistinguishable for any γ over V.

Definition 8. *Let $V \subseteq At$ be a vocabulary and $T \subseteq V$ be a set of atoms. Then, the* V-T-denotation *of a fork F is a T-view defined as follows:*

$$\langle\!\langle F \rangle\!\rangle^T_V \stackrel{\text{def}}{=} {}^{\downarrow}\!\left\{ \mathcal{H}_V \;\middle|\; \mathcal{H} \in \langle\!\langle F \rangle\!\rangle^{T'} \textit{ s.t. } T' \cap V = T \textit{ and } \mathcal{H} \textit{ is not } V\textit{-vacuous} \right\}$$

In other words, we collect all the non-vacuous supports $\mathcal{H}$ that belong to any T'-denotation $\langle\!\langle F \rangle\!\rangle^{T'}$ such that T' coincides with T for atoms in V, and then we

[3] This notion is analogous to condition *ii*) in the definition of V-SE-models that characterises relativised strong equivalence [17].

project the supports taking $\mathcal{H}_V$. In doing so, we can just consider maximal $\mathcal{H}$'s in $\langle\!\langle F \rangle\!\rangle^{T'}$ (and add the empty T-support $[\,]$ to the result if necessary). As might be expected, projecting the T-denotation of a fork F on a superset $V \supseteq At(F)$ of its atoms produces no effect:

Proposition 11. *For any vocabulary $V \subseteq At$, fork F with $At(F) \subseteq V$ and set $T \subseteq V$ of atoms, $\langle\!\langle F \rangle\!\rangle^T_V = \langle\!\langle F \rangle\!\rangle^T$.*

More interestingly, the V-T-denotation of F can be used to precisely characterise its projected stable models.

Proposition 12. *For any vocabulary $V \subseteq At$, fork F and set $T \subseteq V$ of atoms, it holds that $T \in SM_V[F]$ iff $\langle\!\langle F \rangle\!\rangle^T_V = \downarrow\![\, T \,]$.*

As a main result, we naturally extend Proposition 9 to the projective case.

Theorem 1. *For any vocabulary $V \subseteq At$, forks F, G, the following hold:*
(i) *$F \mathrel{|\!\!\sim}_V G$ iff $\langle\!\langle F \rangle\!\rangle^T_V \subseteq \langle\!\langle G \rangle\!\rangle^T_V$ for every set $T \subseteq V$ of atoms, and*
(ii) *$F \cong_V G$ iff $\langle\!\langle F \rangle\!\rangle^T_V = \langle\!\langle G \rangle\!\rangle^T_V$ for every set $T \subseteq V$ of atoms.*

Moreover, the following result shows that we can just use a formula as a context instead of an arbitrary fork.

Proposition 13. *For any vocabulary $V \subseteq At$, forks F, G, the following hold:*
(i) *$F \mathrel{|\!\!\sim}_V G$ iff $SM[F \wedge \gamma] \subseteq SM[G \wedge \gamma]$ for any formula γ with $At(\gamma) \subseteq V$*
(ii) *$F \cong_V G$ iff $SM[F \wedge \gamma] = SM[G \wedge \gamma]$ for any formula γ with $At(\gamma) \subseteq V$.*

As an immediate consequence we can extend the characterisation of PSE from disjunctive logic programs in [12] to arbitrary propositional formulas.

Corollary 3. *For any vocabulary $V \subseteq At$, formulas φ, ψ, the following hold:*
(i) *$\varphi \mathrel{|\!\!\sim}_V \psi$ iff $\langle\!\langle \varphi \rangle\!\rangle^T_V \subseteq \langle\!\langle \psi \rangle\!\rangle^T_V$ for every set $T \subseteq V$ of atoms, and*
(ii) *$\varphi \cong_V \psi$ iff $\langle\!\langle \varphi \rangle\!\rangle^T_V = \langle\!\langle \psi \rangle\!\rangle^T_V$ for every set $T \subseteq V$ of atoms.*

As we mentioned in the introduction, PSE is closely related to forgetting. Let At be a signature and $V \subseteq At$. Given a expression φ over At, a *forgetting operator* is a partial function $\mathtt{f}(\varphi, V) = \psi$ that assigns a new expression ψ over[4] V. Operator $\mathtt{f}$ is said to be *strongly persistent* iff $\varphi \cong_V \mathtt{f}(\varphi, V)$ for every formula φ and set $V \subseteq At$ of atoms for which it is defined. Now, imagine we wish to apply forgetting on a fork F over At keeping atoms $V \subseteq At$. In light of Corollary 3, we can start by obtaining the projected denotations $\langle\!\langle F \rangle\!\rangle^T_V$ for all $T \subseteq V$. This corresponds to a set of T-views that can be precisely captured by another fork over V, as stated by the following result.

Proposition 14. *Given a vocabulary $V \subseteq At$, and assignment σ so that $\sigma(T)$ is some arbitrary T-view for each $T \subseteq V$, there exists a fork G such that $At(G) \subseteq V$ and $\langle\!\langle G \rangle\!\rangle^T = \sigma(T)$ for all $T \subseteq V$.*

[4] Note that we are defining the forgetting operator with respect to the projected signature instead of the forgotten atoms $At \setminus V$.

In other words, we can always define $\mathtt{f}(F,V) \stackrel{\text{def}}{=} G$ by obtaining a fork G over V s.t. $\langle\!\langle\, G \,\rangle\!\rangle^T = \langle\!\langle\, F \,\rangle\!\rangle^T_V$ and, by Proposition 11 and Theorem 1, $G \cong_V F$.

Theorem 2. *For every fork F and set $V \subseteq At$ of atoms, there is a fork G such that $At(G) \subseteq V$ and $F \cong_V G$. Consequently, there is a total, strongly persistent forgetting operator over forks.*

Recall from [10], that such a total operator has been shown not to exist for HT. However, since every propositional formula is also a fork, forgetting in HT is now possible if we allow the target language to be extended with the fork '$\,|\,$' operator – "we can always forget as a fork." Furthermore, the following relation between the T-denotation of forks and of formulas sheds light to the reason why it is not possible to forget inside HT.

Proposition 15. *Given sets $T \subseteq V \subseteq At$ of atoms, then:*
(i) *any formula φ with $At(\varphi) \subseteq V$ satisfies $\langle\!\langle\, \varphi \,\rangle\!\rangle^T_V = \downarrow\![\![\, \varphi \,]\!]^T$ and, thus, $\langle\!\langle\, \varphi \,\rangle\!\rangle^T_V$ has a $\preceq$-maximum element;*
(ii) *for every T-view Δ with a $\preceq$-maximum element, there is a propositional formula φ with $At(\varphi) \subseteq V$ that satisfies $\langle\!\langle\, \varphi \,\rangle\!\rangle^T_V = \Delta$ and $\langle\!\langle\, \varphi \,\rangle\!\rangle^{T'}_V = \{[\,]\}$ for every $T' \subseteq V$ with $T' \neq T$.*

That is, there is a one-to-one correspondence between each assignment of T-views with some $\preceq$-maximum (i.e. *unique* maximal) element and a formula[5] modulo strong equivalence in the vocabulary V. On the other hand, there are T-views that have more than one $\preceq$-maximal element and, thus, they *cannot be represented* as formulas. That is, there are more theories modulo $\cong_V$ than theories over V modulo $\cong$.

As an example, consider program P_m from the introduction interpreted as the conjunction of its rules, and assume we want to forget ma and mb. Let us take all subsets $T' \subseteq \{a, b, ma, mb\}$. Since P_m is a formula, all its T'-views will have a $\preceq$-maximum element, $[\![\, P_m \,]\!]^{T'}$, shown in the left table of Figure 1 where, for brevity, we only show cases of non-empty supports (i.e., when T' is a classical model). Observe that none of these supports is V-vacuous because we never get a strictly smaller $H \subset T$ that coincides with T in $V = \{a, b\}$. Now, according to the definition of $\langle\!\langle\, P_m \,\rangle\!\rangle^T_V$, for each $T \subseteq V = \{a, b\}$ we must find those T' such that $T' \cap \{a, b\} = T$. For $T = \{a\}$ the only possibility is $T' = \{ma, a\}$ that, after removing ma, yields a maximum support $[\{a\}]$. The case for $T = \{b\}$ is completely symmetric, yielding maximum support $[\{b\}]$. But for $T = \{a, b\}$ we get three candidate interpretations, $T'_1 = \{ma, a, b\}$, $T'_2 = \{mb, a, b\}$ and $T'_3 = \{ma, mb, a, b\}$. A first observation is that the support for $T'_3 = \{ma, mb, a, b\}$ is $\{a, b\}$-vacuous, since it contains $\{ma, a, b\}$ and $\{mb, a, b\}$ that coincide with T'_3 for atoms $\{a, b\}$. After removing ma, mb in the supports of the non-vacuous candidates, T'_1 and T'_2, the respective results are $[\ \{a, b\}\ \{a\}\ ]$ and $[\ \{a, b\}\ \{b\}\ ]$ that are not $\preceq$-comparable. Therefore, they become the two $\preceq$-maximal supports in the T-view

[5] A way to obtain the formula is collecting the set of HT-countermodels (those not in the T-supports) and applying the method in [18] to get a minimal logic program.

$\langle\!\langle P_m \rangle\!\rangle_V^{\{a,b\}}$. Proposition 15 tells us that this *is not representable* as a propositional formula, although Proposition 14 always guarantees a representation as a fork. In particular, this set of T-views can be captured by the fork $(a \mid b)$ as we expected from the discussion in the introduction. Analogously, forgetting fa, fb in P_f yields a second fork $(a \mid b)$. Thus, the whole program $P_2 = P_m \wedge P_f$ is $\{a,b\}$-strongly equivalent to $(a \mid b) \wedge (a \mid b)$ that, as we discussed in Section 4, amounts to $(a \mid b \mid a \wedge b)$. It is easy to see that the (non-empty) T-denotations of this fork are:

T	maximal supports in $\langle\!\langle P_2 \rangle\!\rangle^T$
$\{a\}$	[$\{a\}$]
$\{b\}$	[$\{b\}$]
$\{a,b\}$	[$\{a,b\}$]

as the three cases are stable models. But this means that each T-view has a $\preceq$-maximum T-support, and so, the fork is representable as a formula. Program P_1 has precisely the same T-views, so it is strongly equivalent to P_2.

T'	max. supports in $\langle\!\langle P_m \rangle\!\rangle^{T'}$
$\{ma,a\}$	[$\{ma,a\}$]
$\{mb,b\}$	[$\{mb,b\}$]
$\{ma,a,b\}$	[$\{ma,a,b\}$ $\{ma,a\}$]
$\{mb,a,b\}$	[$\{mb,a,b\}$ $\{mb,b\}$]
$\{ma,mb,a,b\}$	[$\{ma,mb,a,b\}$ $\{mb,a,b\}$ $\{mb,b\}$ $\{ma,a,b\}$ $\{ma,a\}$]

T	max. supports in $\langle\!\langle P_m \rangle\!\rangle_V^T$
$\{a\}$	[$\{a\}$]
$\{b\}$	[$\{b\}$]
$\{a,b\}$	[$\{a,b\}$ $\{b\}$] [$\{a,b\}$ $\{a\}$]

Fig. 1. Forgetting ma and mb in P_m.

To illustrate a case not requiring forks, take a second example $\alpha = (\neg p \to q) \wedge (\neg q \to p)$ and assume we want to forget q. The maximal (non-empty) supports in the views of α are:

T'	maximal supports in $\langle\!\langle \alpha \rangle\!\rangle^{T'}$
$\{p\}$	[$\{p\}$]
$\{q\}$	[$\{q\}$]
$\{p,q\}$	[$\{p,q\}$ $\{p\}$ $\{q\}$ $\emptyset$]

Note now that the last support is $\{p\}$-vacuous because $\{p\} \subset \{p,q\} = T'$ but they coincide in the truth of $V = \{p\}$. Thus, for $T = \{p\}$ we only have the non-vacuous $T' = \{p\}$ and we get $[\{p\}]$ as maximum support. For $T = \emptyset$ we have $T' = \{q\}$ and, after removing atom q, we obtain $[\emptyset]$ as maximum support. As both views have maximum supports, this is representable as a propositional formula, that in this case corresponds to the choice $p \vee \neg p$ or, if preferred, $\neg\neg p \to p$.

6 Related work and conclusions

We have extended the syntax and semantics of Here-and-There (HT) to deal with a new type of construct '|' called *fork*. We have studied the property of *projective strong equivalence* (PSE) for forks: two forks satisfy PSE for a vocabulary V iff they yield the same stable models projected on V for any context over V. We also provided a semantic characterisation of PSE that allowed us to prove that it is always possible to forget (under strong persistence) an auxiliary atom in a fork, something recently proved to be false in standard HT [10].

Our work on PSE is strongly related to [12] as, in fact, PSE is one of the cases studied in that paper, which was focused on disjunctive logic programs. The structures used there, called *certificates*, correspond to our non-empty T-supports, while allowing empty T-supports or the extension to T-views as ideals were needed here in order to fulfill our goal of providing an algebraic semantics for forks. Using this relation, our Corollary 3 extends the result in [12] to arbitrary formulas in HT and this is still valid for certificates. Previous complexity results can be used to prove that brave and cautious reasoning with forks in normal form are Σ_2^{P} and Π_2^{P}-complete, respectively. Similarly, checking PSE of forks in normal form is Π_4^{P}-complete. These results follow from the fact that every fork in normal form can be replaced (under PSE) by some (log-space constructible) propositional formula using new auxiliary atoms. We also conjecture that these results hold for arbitrary forks, but the formal proof is left for future work.

For future work, we plan to extend these results to other characterisations of equivalence [19] and, in particular, study the case of Projective Uniform Equivalence, that is, PSE for vocabulary V where the context theories are sets of atoms from V. Another natural extension of forks is to consider the addition of probabilities. In that way, for instance, our example about Mendelian laws could reflect the proportion of each possible combination, 1/4 for $\{a\}$ and $\{b\}$ and 1/2 for $\{a, b\}$. Doing so, we conjecture a strong formal connection to *CP-logic* [20], where the use of disjunction behaves as our fork connective. Note that, although forks do not deal with probabilities, they allow a more general syntax than CP-logic programs, which additionally require the well-founded model to be defined on all atoms. Similarly, we also plan a formal comparison with non-deterministic causal laws [21].

Acknowledgements. We are especially thankful to the anonymous reviewers for their helpful suggestions to improve the paper.

References

1. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
2. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artificial Intelligence **187-188** (2012) 52–89
3. Faber, W., Pfeifer, G., Leone, N., Dell'Armi, T., Ielpa, G.: Design and implementation of aggregate functions in the DLV system. Theory and Practice of Logic Programming **8**(5-6) (2008) 545–580

4. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Magazine **37**(3) (2016) 53–68
5. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R., Bowen, K., eds.: Proceedings of the 19th International Conference and Symposium of Logic Programming (ICLP'88), MIT Press (1988) 1070–1080
6. Pearce, D.: A new logical characterisation of stable models and answer sets. In Dix, J., Pereira, L.M., Przymusinski, T.C., eds.: Selected Papers from the Non-Monotonic Extensions of Logic Programming (NMELP'96). Volume 1216 of Lecture Notes in Artificial Intelligence., Springer-Verlag (1996) 57–70
7. Pearce, D.: Equilibrium logic. Annals of Mathematics and Artificial Intelligence **47**(1-2) (2006) 3–41
8. Ferraris, P., Lee, J., Lifschitz, V.: A new perspective on stable models. In Sangal, R., Mehta, H., Bagga, R.K., eds.: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07), Morgan Kaufmann Publishers Inc. (2007) 372–379
9. Harrison, A., Lifschitz, V., Truszczynski, M.: On equivalence of infinitary formulas under the stable model semantics. Theory and Practice of Logic Programming **15**(1) (2015) 18–34
10. Gonçalves, R., Knorr, M., Leite, J.: You can't always forget what you want: On the limits of forgetting in answer set programming. In Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F., eds.: Proceedings of 22nd European Conference on Artificial Intelligence (ECAI'16). Volume 285 of Frontiers in Artificial Intelligence and Applications., IOS Press (2016) 957–965
11. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2**(4) (2001) 526–541
12. Eiter, T., Tompits, H., Woltran, S.: On solution correspondences in answer-set programming. In Kaelbling, L.P., Saffiotti, A., eds.: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05), Professional Book Center (2005) 97–102
13. Cabalar, P., Pearce, D., Valverde, A.: Reducing propositional theories in equilibrium logic to logic programs. In Bento, C., Cardoso, A., Dias, G., eds.: Proceedings of the 12th Portuguese Conference on Progress in Artificial Intelligence (EPIA'05). Volume 3808 of Lecture Notes in Computer Science., Springer (2005) 4–17
14. Ferraris, P.: Answer sets for propositional theories. In Baral, C., Greco, G., Leone, N., Terracina, G., eds.: Proc. of the 8th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'05). Volume 3662 of Lecture Notes in Computer Science. Springer (2005) 119–131
15. Aguado, F., Cabalar, P., Pearce, D., Pérez, G., Vidal, C.: A denotational semantics for equilibrium logic. Theory and Practice of Logic Programming **15**(4-5) (2015) 620–634
16. Cabalar, P., Ferraris, P.: Propositional theories are strongly equivalent to logic programs. Theory and Practice of Logic Programming **7**(6) (2007) 745–759
17. Eiter, T., Fink, M., Woltran, S.: Semantical characterizations and complexity of equivalences in answer set programming. ACM Transactions on Computational Logic **8**(3) (2007) 17
18. Cabalar, P., Pearce, D., Valverde, A.: Minimal logic programs. In Dahl, V., Niemelä, I., eds.: Proceedings of the 23rd International Conference on Logic Programming, (ICLP'07), Springer (2007) 104–118

19. Fink, M.: A general framework for equivalences in answer-set programming by countermodels in the logic of here-and-there. Theory and Practice of Logic Programming **11**(2-3) (2011) 171–202
20. Vennekens, J., Denecker, M., Bruynooghe, M.: CP-logic: A language of causal probabilistic events and its relation to logic programming. Theory and Practice of Logic Programming **9**(3) (2009) 245–308
21. Bochman, A.: On disjunctive causal inference and indeterminism. In: Proceedings of The Workshop on Nonmonotonic Reasoning, Actions and Change (NRAC'03). (2003) 45–50