Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Journal Track

# Forgetting Auxiliary Atoms in Forks (Extended Abstract)*

**Felicidad Aguado**[1], **Pedro Cabalar**[1], **Jorge Fandinno**[2], **David Pearce**[3], **Gilberto Pérez**[1] and **Concepción Vidal**[1]
[1]University of Corunna, Spain
[2]University of Potsdam, Germany
[3]Universidad Politécnica de Madrid, Spain
{aguado, cabalar, gilberto.pvega, concepcion.vidalm}@udc.es,
fandinno@uni-potsdam.de, david.pearce@upm.es

## Abstract

This work tackles 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 [Baral, 2003]) 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 [Gebser *et al.*, 2012; Faber *et al.*, 2008] and application domains [Erdem *et al.*, 2016], but also from its solid theoretical foundations, rooted in the *stable models* [Gelfond and Lifschitz, 1988] semantics for normal logic programs that was later generalised to arbitrary propositional [Pearce, 1996], first-order [Pearce, 2006; Ferraris *et al.*, 2007] and infinitary [Harrison *et al.*, 2015] formulas. An important breakthrough that supported these extensions of ASP has been its logical characterisation in terms of *Equilibrium Logic* [Pearce, 1996] and its monotonic basis, the intermediate logic of *Here-and-There* (HT). Despite its expressiveness, a recent result [Gonçalves *et al.*, 2016] has shown that Equilibrium Logic has limitations in capturing the representational power of auxiliary atoms, which cannot always be forgotten.

In this paper, we extend logic programs to include a new construct '|' we call *fork* and 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. We also review the notion of *projective strong equivalence* (PSE), that is, whether two programs are equivalent over a restricted vocabulary in any context that uses also the same restricted vocabulary [Eiter *et al.*, 2005]. Finally, we show that using this construct is always possible to forget any set of atoms, something that was not possible without it.

## 2 Motivation Example

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 $\{a, b\}$. These are, in fact, the three classical models of disjunction $a \vee b$. To obtain these three solutions as stable models in ASP, the straightforward way would be to use the three rules:

$$a \vee \neg a \qquad b \vee \neg b \qquad \bot \leftarrow \neg a \wedge \neg b \qquad (P_1)$$

We assume here some familiarity with ASP: disjunctions of the form $p \vee \neg p$ act as non-deterministic *choice rules* (allowing the arbitrary inclusion of atom $p$) and $\bot \leftarrow \neg a \wedge \neg b$ is a *constraint* forbidding models where $a \vee b$ does not hold. Moreover, when we include $p \vee \neg p$ for all atoms, as in the example, stable models just coincide with classical models. 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,

*This paper is an extended abstract of an article entitled *Forgetting Auxiliary Atoms in Forks* and published in Artificial Intelligence [Aguado *et al.*, 2019].

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 $P_2$:

$$\begin{array}{llll} ma \vee mb & a \leftarrow ma & b \leftarrow mb & (P_m) \\ fa \vee fb & a \leftarrow fa & b \leftarrow fb & (P_f) \end{array}$$

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). As mentioned before, we call this notion *projective strong equivalence* w.r.t. $V$, or *$V$-strong equivalence* for short.

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 $P_f$).

## 3 Forks and $T$-views

Let $At$ be a finite set of atoms called the *(propositional) signature*. A *(propositional) formula* $\varphi$ is defined using the grammar:

$$\varphi \ ::= \ \bot \ \mid \ p \ \mid \ \varphi \wedge \varphi \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \rightarrow \varphi$$

where $p$ is an atom $p \in At$. We use Greek letters $\varphi, \psi, \gamma$ and their variants to stand for formulas. A *fork* is defined using the grammar:

$$F \ ::= \ \bot \ \mid \ p \ \mid \ (F \mid F) \ \mid \ F \wedge F \ \mid \ \varphi \vee \varphi \ \mid \ \varphi \rightarrow F$$

where $\varphi$ is a propositional formula and $p \in At$ is an atom. We use Greek letters $F, G, H$ and their variants to stand for forks. We define the derived operators $\neg\varphi \stackrel{\text{def}}{=} (\varphi \rightarrow \bot)$ and $\top \stackrel{\text{def}}{=} \neg\bot$. Notice that the fork operator '|' cannot occur in the scope of disjunction or negation, since $\neg F$ would stand for $F \rightarrow \bot$ and implications do not allow forks in the antecedent. This restricted syntax suffices for the purposes of the current paper.

The semantics of forks is defined in terms of denotations in two steps. First we define the denotation of propositional formulas and then we extend it to arbitrary forks.

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 $\mathcal{H} \neq \emptyset$. We write $\mathbf{H}_T$ to stand for the set of all possible $T$-supports. To increase readability of examples, we 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 $\varphi$, that is, the set of $H$'s such that $\langle H, T\rangle \models \varphi$ in here-and-there logic. 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, thus, $\langle T, T\rangle$ must be a model too; this is why we require $T \in \mathcal{H}$ in the set.

**Definition 1 ($T$-denotation; Aguado *et al.* 2015)** *Let $T \subseteq At$. The* $T$-denotation *of a formula $\varphi$, written $[\![\,\varphi\,]\!]^T$, is a $T$-support recursively defined as follows:*

$$\begin{array}{rcl} [\![\,\bot\,]\!]^T & \stackrel{\text{def}}{=} & [\,] \\ [\![\,p\,]\!]^T & \stackrel{\text{def}}{=} & \{H \subseteq T \mid p \in H\} \\ [\![\,\varphi \wedge \psi\,]\!]^T & \stackrel{\text{def}}{=} & [\![\,\varphi\,]\!]^T \cap [\![\,\psi\,]\!]^T \\ [\![\,\varphi \vee \psi\,]\!]^T & \stackrel{\text{def}}{=} & [\![\,\varphi\,]\!]^T \cup [\![\,\psi\,]\!]^T \end{array}$$

$$[\![\varphi \rightarrow \psi]\!]^T \stackrel{\text{def}}{=} \begin{cases} [\,] & \text{if } [\![\varphi]\!]^T \neq [\,] \text{ and } [\![\psi]\!]^T = [\,] \\ \overline{[\![\varphi]\!]^T} \cup [\![\psi]\!]^T & \text{otherwise} \end{cases}$$

$T$-denotations satisfy that $H \in [\![\,\varphi\,]\!]^T$ iff $\langle H, T\rangle \models \varphi$ in the logic of here-and-there. Interestingly, when not empty, the fewer models in $[\![\,\varphi\,]\!]^T$, the more supported is $T$, since it is closer to being stable. Seeing "more supported" as an ordering relation, the "most supported" $[\![\,\varphi\,]\!]^T$ (the top element) would precisely be $[\![\,\varphi\,]\!]^T = [\ T\ ]$ corresponding to a stable model. This ordering relation is formally defined below.

**Definition 2** *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 $\mathcal{H} \supseteq \mathcal{H}' \neq [\,]$.*

The relation $\preceq_T$ is a partial order on $\mathbf{H}_T$ with $[\,]$ and $[\ T\ ]$ its bottom and top elements, respectively. 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.

**Example 2 (Implementing a choice)** *To implement a choice rule for atom $p$ (in modern ASP syntax, written $\{p\}$) a knowledge engineer uses an auxiliary atom $q$. As a first option, she considers the use of rules:*

$$(\neg p \rightarrow q) \wedge (\neg q \rightarrow p) \tag{1}$$

*However, having a disjunctive ASP solver, another possibility could be:*

$$p \vee q \tag{2}$$

*Is there any substantial difference?* □

| $T$ | $[\![ (1) ]\!]^T$ | $[\![ (2) ]\!]^T$ |
|---|---|---|
| $\emptyset$ | $[\,]$ | $[\,]$ |
| $\{p\}$ | $[\,\{p\}\,]$ | $[\,\{p\}\,]$ |
| $\{q\}$ | $[\,\{q\}\,]$ | $[\,\{q\}\,]$ |
| $\{p,q\}$ | $[\{p,q\}\ \{p\}\ \{q\}\ \emptyset\,]$ | $[\,\{p,q\}\ \{p\}\ \{q\}\,]$ |

Figure 1: $T$-denotations for (1) and (2).

The $T$-denotations of both options are shown in Figure 1. As mentioned above, stable models correspond to $T$'s such that $[\![ \varphi ]\!]^T = [\,T\,]$, that is, we get the two stable models $\{p\}$ and $\{q\}$ in both columns. In fact, all rows coincide except for $T = \{p,q\}$ where $\emptyset \in [\![ (1) ]\!]^{\{p,q\}}$ but $\emptyset \notin [\![ (2) ]\!]^{\{p,q\}}$. A difference in $T$-denotations means that formulas are not HT equivalent, and so, they are not strongly equivalent [Lifschitz *et al.*, 2001]. In fact, the counterexample of strong equivalence is well-known in the literature: adding $(p \to q) \wedge (q \to p)$ to (1) yields no stable model, while the same addition to (2) produces stable model $\{p,q\}$.

Let us now introduce the semantics of arbitrary forks. Given a $T$-support $\mathcal{H}$ we define the set of (non-empty) $\preceq$-smaller supports $\downarrow\mathcal{H} = \{\mathcal{H}' \mid \mathcal{H}' \preceq \mathcal{H}\} \setminus \{\,[\,]\,\}$. This is usually called the *ideal* of $\mathcal{H}$. Note that, the empty support $[\,]$ is not included in the ideal. As a result, $\downarrow[\,] = \emptyset$. We extend this notation to any set of supports $\Delta$ so that $^{\downarrow}\Delta \stackrel{\text{def}}{=} \{\,\mathcal{H}' \mid \mathcal{H}' \preceq \mathcal{H}, \mathcal{H} \in \Delta\,\} \setminus \{\,[\,]\,\}$.

**Definition 3 ($T$-view)** *A $T$-*view *is a set of $T$-supports $\Delta \subseteq \mathbf{H}_T$ that is $\preceq$-closed, i.e., $^{\downarrow}\Delta = \Delta$.*

If $\Delta$ is a $T$-view and the $\preceq$-greatest $T$-support $[\,T\,]$ is included in $\Delta$, then $\Delta$ is precisely $\downarrow[\,T\,]$. We proceed next to define the semantics of forks in terms of $T$-views. To emphasize the duality between conjunction and disjunction, we will be interested in dealing with a weaker version of the membership relation, $\hat{\in}$, defined as follows. Given a $T$-view $\Delta$, we write $\mathcal{H} \,\hat{\in}\, \Delta$ iff $\mathcal{H} \in \Delta$ or both $\mathcal{H} = [\,]$ and $\Delta = \emptyset$. We are now ready to extend the concept of $T$-denotation to forks.

**Definition 4 ($T$-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:*

$$\langle\!\langle\, \bot \,\rangle\!\rangle^T \stackrel{\text{def}}{=} \emptyset$$

$$\langle\!\langle\, p \,\rangle\!\rangle^T \stackrel{\text{def}}{=} \downarrow[\![ p ]\!]^T \quad \textit{for any atom } p$$

$$\langle\!\langle F \wedge G \rangle\!\rangle^T \stackrel{\text{def}}{=} {}^{\downarrow}\{\, \mathcal{H} \cap \mathcal{H}' \mid \mathcal{H} \in \langle\!\langle F \rangle\!\rangle^T \textit{ and } \mathcal{H}' \in \langle\!\langle G \rangle\!\rangle^T \,\}$$

$$\langle\!\langle \varphi \vee \psi \rangle\!\rangle^T \stackrel{\text{def}}{=} {}^{\downarrow}\{\, \mathcal{H} \cup \mathcal{H}' \mid \mathcal{H} \,\hat{\in}\, \langle\!\langle\, \varphi \,\rangle\!\rangle^T \textit{ and } \mathcal{H}' \,\hat{\in}\, \langle\!\langle\, \psi \,\rangle\!\rangle^T \,\}$$

$$\langle\!\langle \varphi \to F \rangle\!\rangle^T \stackrel{\text{def}}{=} \begin{cases} \{\, 2^T \,\} & \textit{if } [\![ \varphi ]\!]^T = [\,] \\ {}^{\downarrow}\{\, \overline{[\![\varphi]\!]^T} \cup \mathcal{H} \mid \mathcal{H} \in \langle\!\langle F \rangle\!\rangle^T \,\} & \textit{otherwise} \end{cases}$$

$$\langle\!\langle F \mid G \rangle\!\rangle^T \stackrel{\text{def}}{=} \langle\!\langle\, F \,\rangle\!\rangle^T \cup \langle\!\langle\, G \,\rangle\!\rangle^T$$

The fork operator '|' is commutative, associative and idempotent. Conjunction and implication distribute over '|'. As for the rest of operators, note that the definitions above also cover propositional formulas. In fact, we have the following relation between $T$-denotations as $T$-views and $T$-supports: $\langle\!\langle\, \varphi \,\rangle\!\rangle^T = \downarrow[\![ \varphi ]\!]^T$ for every propositional formula $\varphi$.

**Definition 5** *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 $F$.*

For any propositional formula, this definition coincides with the standard definition of stable models. The intuition behind a fork is that we can collect its stable models independently, that is, $SM[F \mid G] = SM[F] \cup SM[G]$.

## 4 Projective Strong Equivalence

Let us now formally define projective strong equivalence for forks. Since propositional formulas are also forks, this definition also applies to them. We represent the projection of the stable models of some fork onto some vocabulary $V$ as $SM_V[\varphi] \stackrel{\text{def}}{=} \{\, T \cap V \mid T \in SM[\varphi] \,\}$.

**Definition 6 (Projective strong equivalence)** *Let $V \subseteq At$ be some vocabulary. A fork $F$ is $V$-*strongly equivalent *to fork $G$, written $F \cong_V G$ if $SM_V[F \wedge L] = SM_V[G \wedge L]$ for any fork $L$ such that $At(L) \subseteq V$.*

When $V \supseteq At(F) \cup At(G)$ we talk about (non-projective) strong equivalence and drop the $V$ subindex. Strong equivalence has a simple characterisation in these terms: $F \cong G$ iff $\langle\!\langle\, F \,\rangle\!\rangle^T = \langle\!\langle\, G \,\rangle\!\rangle^T$ for every $T \subseteq At$.

Providing a semantic characterisation of projective strong equivalence $\cong_V$ requires some more notation. We define $\mathcal{H}_V$ as the projection of every set in $\mathcal{H}$ to the vocabulary $V$, i.e., $\mathcal{H}_V \stackrel{\text{def}}{=} \{H \cap V \mid H \in \mathcal{H}\}$. A $T$-support $\mathcal{H}$ is *$V$-unfeasible* iff there is some $H \subset T$ in $\mathcal{H}$ satisfying $H \cap V = T \cap V$; it is *$V$-feasible* otherwise. The reason for the name "unfeasible" is that, if we take a formula $\varphi$ with denotation $[\![ \varphi ]\!]^T = \mathcal{H}$, then $T$ will never become stable if we are only allowed to use vocabulary $V$ when adding a context $\gamma$.

**Definition 7** *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, denoted as $\langle\!\langle\, F \,\rangle\!\rangle^T_V$, is defined as follows:*

$$^{\downarrow}\{\, \mathcal{H}_V \,|\, \mathcal{H} \in \langle\!\langle F \rangle\!\rangle^{T'} \textit{ s.t. } T' \cap V = T \textit{ and } \mathcal{H} \textit{ is } V\textit{-feasible} \,\}$$

In other words, $\langle\!\langle\, F \,\rangle\!\rangle^T_V$ collects all the feasible 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 project the supports taking $\mathcal{H}_V$. In doing so, we can just consider maximal $\mathcal{H}$'s in $\langle\!\langle\, F \,\rangle\!\rangle^{T'}$. 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, that is, $\langle\!\langle\, F \,\rangle\!\rangle^T_V = \langle\!\langle\, F \,\rangle\!\rangle^T$. More interestingly, the $V$-$T$-denotation of $F$ can be used to precisely characterise its projected stable models and projective strong equivalence.

**Theorem 1** *For any vocabulary $V \subseteq At$, set of atoms $T \subseteq V$ and forks $F, G$, the following holds:*

(i) $T \in SM_V[F]$ *iff* $\langle\!\langle\, F \,\rangle\!\rangle^T_V = \downarrow[\,T\,]$.

(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.*

| $T$ | maximal supports in $\langle\!\langle\, P_m \,\rangle\!\rangle_V^T$ |
|---|---|
| $\{a\}$ | $[\,\{a\}\,]$ |
| $\{b\}$ | $[\,\{b\}\,]$ |
| $\{a,b\}$ | $[\,\{a,b\}\,\{b\}\,] \quad [\,\{a,b\}\,\{a\}\,]$ |

Figure 2: Forgetting $ma$ and $mb$ in $P_m$.

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

Figure 3: Some $T$-views of $P_m$.

**Example 3 (Ex. 2 continued)** We can now check the PSE of $(1) = (\neg p \to q) \wedge (\neg q \to p)$ and $(2) = (p \vee q)$. To do so, we have to test if $\langle\!\langle\, (1) \,\rangle\!\rangle_V^T = \langle\!\langle\, (2) \,\rangle\!\rangle_V^T$ for every $T \subseteq V$ and any vocabulary $V$ not containing $q$. Since $[\![\, (1) \,]\!]^T = [\![\, (2) \,]\!]^T$ for all but $T = \{p,q\}$ we can restrict ourselves to the study to $At = \{p,q\}$ and $V = \{p\}$. Take $\langle\!\langle\, (1) \,\rangle\!\rangle_V^T$ first. Since (1) is a propositional formula (it contains no forks), we have $\langle\!\langle\, (1) \,\rangle\!\rangle^{T'} = \downarrow [\![\, (1) \,]\!]^{T'}$, that is, the maximal supports are just the $T$-denotations we already obtained in Figure 1. Now, for vocabulary $V = \{p\}$ we may only have $T = \{p\}$ and $T = \emptyset$. For the first case, the potential candidates $T'$ such that $T' \cap V = \{p\}$ are the rows $T' = \{p\}$ and $T' = \{p,q\}$ from Figure 1. However, the support we have for the latter is $V$-unfeasible because it contains $H = \{p\}$ with $H \subset \{p,q\} = T'$ and $H \cap V\{p\} = T' \cap V$. Thus, for $T = \{p\}$ we only have the feasible (maximum) support $[\{p\}]$ from $T' = \{p\}$ which yields $\langle\!\langle\, (1) \,\rangle\!\rangle_V^T = \downarrow [\,\{p\}\,]$. For $T = \emptyset$ the only non-empty case is $T' = \{q\}$ and, after removing atom $q$, we obtain $[\emptyset]$ as maximum support, i.e., $\langle\!\langle\, (1) \,\rangle\!\rangle_V^\emptyset = \downarrow [\,\emptyset\,]$. It is easy to see that (2) yields the same result: the only difference we have is in the last row of Figure 1, for $T' = \{p,q\}$, but this support is $V$-unfeasible again, as it also contains $H = \{p\}$. To sum up $\langle\!\langle\, (1) \,\rangle\!\rangle_V^T = \langle\!\langle\, (2) \,\rangle\!\rangle_V^T = \downarrow [\,\{p\}\,]$ and $\langle\!\langle\, (1) \,\rangle\!\rangle_V^\emptyset = \langle\!\langle\, (2) \,\rangle\!\rangle_V^\emptyset = \downarrow [\,\emptyset\,]$ so both formulas satisfy PSE, that is, $(1) \cong_V (2)$.

## 5 Forgetting

We show now how the addition of forks allows us to define a strong persistent forgetting operator that is applicable to any fork and, thus, to any formula. As mentioned in the introduction, this is not possible without the fork operator. Formally, given a $T$-support $\mathcal{H}$, we denote by $\Phi_\mathcal{H}$ a propositional formula such that $[\![\, \Phi_\mathcal{H} \,]\!]^T = \mathcal{H}$. Such a formula always exists and the process to build it was studied by Cabalar and Ferraris [2007].

**Definition 8 (Forgetting operator)** *The fork-forgetting operator* $\mathtt{fk}$ *is a function mapping forks and sets of atoms to forks such that*

$$\mathtt{fk}(F,V) \quad \stackrel{\text{def}}{=} \quad \Phi_{\mathcal{H}_1} \mid \cdots \mid \Phi_{\mathcal{H}_n}$$

*with* $\{\mathcal{H}_1, \ldots \mathcal{H}_n\}$ *being the set of* $\preceq_T$*-maximal* $T$*-supports in* $\langle\!\langle\, F \,\rangle\!\rangle_V^T$ *for all* $T \subseteq V$. □

Operator $\mathtt{fk}$ is a total, strongly persistent forgetting operator over forks. This is formalised as follows.

**Theorem 2** *For every fork* $F$ *and set* $V \subseteq At$ *of atoms,* $At(\mathtt{fk}(F,V)) \subseteq V$ *and* $F \cong_V \mathtt{fk}(F,V)$. □

**Example 4 (Ex. 1 continued)** Consider program $P_m$ from the introduction interpreted as the conjunction of its rules, and assume we want to forget $ma$ and $mb$. The table in Figure 2 shows the $T$-$V$-denotation of $P_m$ for each $T \subseteq V = \{a,b\}$. Cases for $T = \{a\}$ and $T = \{b\}$ are easy to compute. For $T = \{a,b\}$ we get three candidate interpretations, $T_1' = \{ma,a,b\}$, $T_2' = \{mb,a,b\}$ and $T_3' = \{ma,mb,a,b\}$. The table in Figure 3 shows the $T$-views of $P_m$ for these $T_i'$. A first observation is that the support for $T_3' = \{ma,mb,a,b\}$ is $\{a,b\}$-unfeasible because 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 feasible 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 last row of Figure 2. It is also easy to check that the $T$-denotations of fork $a \mid b$ exactly coincide with the one shown in Figure 2 for $P_m$. Therefore, $P_m \cong_V (a \mid b)$. It also can be checked that, after some simplifications, the forgetting operator, $\mathtt{fk}(P_m, V)$, precisely produces the fork $(a \mid b)$.

## 6 Conclusions

We extended the syntax and semantics of Here-and-There (HT) to deal with a new type of construct '|' called *fork*. We 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.

## Acknowledgements

This work was partially supported by MINECO (TIN2017-84453-P), Xunta de Galicia (GPC ED431B 2019/03) and the Salvador de Madariaga programme, Spain; by the European Regional Development Fund (ERDF). The third author is supported by the Alexander von Humboldt Foundation.

## References

[Aguado *et al.*, 2015] Felicidad Aguado, Pedro Cabalar, David Pearce, Gilberto Pérez, and Concepción Vidal. A denotational semantics for equilibrium logic. *Theory and Practice of Logic Programming*, 15(4-5):620–634, 2015.

[Aguado *et al.*, 2019] Felicidad Aguado, Pedro Cabalar, Jorge Fandinno, David Pearce, Gilberto Pérez, and Concepción Vidal. Forgetting auxiliary atoms in forks. *Artif. Intell.*, 275:575–601, 2019.

[Baral, 2003] Chitta Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[Cabalar and Ferraris, 2007] Pedro Cabalar and Paolo Ferraris. Propositional theories are strongly equivalent to logic programs. *Theory and Practice of Logic Programming*, 7(6):745–759, 2007.

[Eiter *et al.*, 2005] Thomas Eiter, Hans Tompits, and Stefan Woltran. On solution correspondences in answer-set programming. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 97–102. Professional Book Center, 2005.

[Erdem *et al.*, 2016] Esra Erdem, Michael Gelfond, and Nicola Leone. Applications of answer set programming. *AI Magazine*, 37(3):53–68, 2016.

[Faber *et al.*, 2008] W. Faber, G. Pfeifer, N. Leone, T. Dell'Armi, and G. Ielpa. Design and implementation of aggregate functions in the DLV system. *Theory and Practice of Logic Programming*, 8(5-6):545–580, 2008.

[Ferraris *et al.*, 2007] Paolo Ferraris, Joohjung Lee, and Vladimir Lifschitz. A new perspective on stable models. In R. Sangal, H. Mehta, and R. K. Bagga, editors, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 372–379. Morgan Kaufmann Publishers Inc., 2007.

[Gebser *et al.*, 2012] M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187-188:52–89, 2012.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proceedings of the 19th International Conference and Symposium of Logic Programming (ICLP'88)*, pages 1070–1080. MIT Press, 1988.

[Gonçalves *et al.*, 2016] Ricardo Gonçalves, Matthias Knorr, and João Leite. You can't always forget what you want: On the limits of forgetting in answer set programming. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, *Proceedings of 22nd European Conference on Artificial Intelligence (ECAI'16)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 957–965. IOS Press, 2016.

[Harrison *et al.*, 2015] Amelia Harrison, Vladimir Lifschitz, and Miroslaw Truszczynski. On equivalence of infinitary formulas under the stable model semantics. *Theory and Practice of Logic Programming*, 15(1):18–34, 2015.

[Lifschitz *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

[Pearce, 1996] David Pearce. A new logical characterisation of stable models and answer sets. In J. Dix, L. Moniz Pereira, and T. C. Przymusinski, editors, *Selected Papers from the Non-Monotonic Extensions of Logic Programming (NMELP'96)*, volume 1216 of *Lecture Notes in Artificial Intelligence*, pages 57–70. Springer-Verlag, 1996.

[Pearce, 2006] David Pearce. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):3–41, 2006.