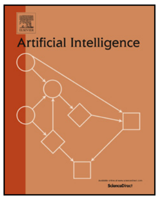
Artificial Intelligence 308 (2022) 103712

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



A polynomial reduction of forks into logic programs

Felicidad Aguado[a,b], Pedro Cabalar[a,*], Jorge Fandinno[c], David Pearce[d], Gilberto Pérez[a,b], Concepción Vidal[a,b]

[a] *University of A Coruña, Spain*
[b] *CITIC Research Center, A Coruña, Spain*
[c] *University of Nebraska, Omaha, NE, USA*
[d] *Universidad Politécnica de Madrid, Spain*

A R T I C L E   I N F O

Article history:
Received 13 March 2021
Received in revised form 22 February 2022
Accepted 24 March 2022
Available online 29 March 2022

Keywords:
Answer set programming
Non-monotonic reasoning
Equilibrium logic
Denotational semantics
Forgetting
Strong equivalence

A B S T R A C T

In this research note we present additional results for an earlier published paper [1]. There, we studied the problem of *projective strong equivalence* (PSE) of logic programs, that is, checking whether two logic programs (or propositional formulas) have the same behaviour (under the stable model semantics) regardless of a common context and ignoring the effect of local auxiliary atoms. PSE is related to another problem called *strongly persistent forgetting* that consists in keeping a program's behaviour after removing its auxiliary atoms, something that is known to be not always possible in Answer Set Programming. In [1], we introduced a new connective '|' called *fork* and proved that, in this extended language, it is always possible to forget auxiliary atoms, but at the price of obtaining a result containing forks. We also proved that forks can be translated back to logic programs introducing new hidden auxiliary atoms, but this translation was exponential in the worst case. In this note we provide a new polynomial translation of arbitrary forks into regular programs that allows us to prove that brave and cautious reasoning with forks has the same complexity as that of ordinary (disjunctive) logic programs and paves the way for an efficient implementation of forks. To this aim, we rely on a pair of new PSE invariance properties.

1. Introduction

Nowadays, *Answer Set Programming* (ASP) [2] is one of the most popular paradigms for practical knowledge representation, reasoning and problem solving. An important part of this success relies on its solid theoretical foundations, rooted in the *stable model* [3] semantics together with its logical formalisations. Among the latter, a prominent approach is the use of *Equilibrium Logic* [4] and its monotonic basis, the intermediate logic of *Here-and-There* (HT), which has been successfully applied to define many different extensions of the stable models semantics. Despite its expressiveness, a result proved in [5] 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 two logic programs from [1]:

* Corresponding author.
E-mail addresses: aguado@udc.es (F. Aguado), cabalar@udc.es (P. Cabalar), jfandinno@unomaha.edu, fandinno@uni-potsdam.de (J. Fandinno), david.pearce@upm.es (D. Pearce), gilberto.pvega@udc.es (G. Pérez), concepcion.vidalm@udc.es (C. Vidal).

https://doi.org/10.1016/j.artint.2022.103712

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

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

Each program, P_m and P_f, is encoding a choice for adding atoms a or b to the respective stable models. To this aim, each program uses its own pair of auxiliary atoms (ma, mb for P_m, and fa, fb for P_f) that allow their respective choices to act *independently* even if the programs are combined together[1] $P_m \wedge P_f$. A natural question is whether $P_m \wedge P_f$ can be replaced by another program P_1 only in terms of atoms a, b, that is, *forgetting* the auxiliary atoms ma, mf, fa and fb, in a way that is "essentially equivalent." More precisely, the kind of equivalence we need would first require that we obtain the same stable models even if we include both programs in a larger arbitrary context Q, that is, we compare $P_m \wedge P_f \wedge Q$ and $P_1 \wedge Q$ for any Q – this is called *strong equivalence* [6]. Moreover, we further need to strengthen strong equivalence by removing auxiliary atoms from the stable models to be compared and forbid their occurrence in the public context Q. This stronger definition corresponds to one of the variants of strong equivalence defined in [7] and it was named in [1] as *Projective Strong Equivalence* (PSE) with respect to some public vocabulary V (or just V*-strongly equivalent* for short). If we take $V = \{a, b\}$, program $P_m \wedge P_f$ is indeed V-strongly equivalent to the program:

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

However, if we take any of the components separately, say just P_m on its own, *there is no possible way to forget its auxiliary atoms* [5] to obtain a program V-strongly equivalent to P_m. Program P_1, for instance, does not work any more: it has a stable model $\{a, b\}$ that cannot be obtained from any of the two stable models, $\{ma, a\}$ and $\{mb, b\}$ of P_m after removing auxiliary atoms. In practice, this impossibility means that auxiliary atoms are more than 'just' auxiliary, as they allow the representation of problems that cannot be captured without them.

In [1], we considered an extension of ASP to cover this lack of expressiveness, introducing a new construct '$|$' called *fork*. Intuitively, the stable models of $(P \mid P')$ correspond to the union of stable models from P and P' in any context Q, that is $SM[(P \mid P') \wedge Q] = SM[P \wedge Q] \cup SM[P' \wedge Q]$. In this extended language, it is always possible to forget auxiliary atoms: for instance, we can represent both P_m and P_f as the V-strongly equivalent fork $(a \mid b)$. As a result, if we forget all auxiliary atoms in $P_m \wedge P_f$ we obtain $(a \mid b) \wedge (a \mid b)$ revealing that the conjunction of forks is not idempotent. In fact, this fork actually amounts to $(a \mid b \mid a \wedge b)$ and has stable models $\{a\}$, $\{b\}$ and $\{a, b\}$. In [1], we provided a denotational semantics that allows one to prove that forgetting is always possible in forks, but some of them, such as $(a \mid b)$, cannot be represented in Equilibrium Logic. We also used this denotational semantics to capture PSE and to characterise those forks that can be equivalently represented as regular formulas.

An open question that remained unanswered in [1] has to do with the complexity of reasoning about forks. In that paper, we showed that there exists a normal form, *unnested forks* (UF), in which fork connectives are not in the scope of another connective. We also provided a polynomial translation from forks in UF normal form into logic programs (adding new fresh auxiliary atoms). As a result, we could prove that the complexity of brave and cautious reasoning for forks *in UF normal form* was the same as in disjunctive logic programs, that is, Σ_2^P and Π_2^P-complete, respectively. For *arbitrary* forks, however, this complexity assessment remained open, since the reduction into UF normal form may cause an exponential blow up due to distributivity laws.

In this research note, we extend the results from [1] by presenting a pair of additional invariance results for PSE that allow us to obtain a polynomial translation of arbitrary forks into regular programs.[2] This new translation is important not only for a future implementation of fork logic programs, but also for proving that brave and cautious reasoning with arbitrary forks has the same complexity as that of ordinary (disjunctive) logic programs.

The rest of the paper is organised as follows. The next section recalls the basic definitions from [1] required to prove the new results, including a revised version of the forks syntax (more comfortable for inductive proofs) together with their denotational semantics. Section 3 revisits the definition of PSE and provides several useful invariance results that will be used for the reduction to logic programs. In Section 4 we present the new reduction and, finally, Section 5 concludes this note.

2. Background

We begin by recalling some basic definitions from the logic of *Here-and-There* [8] (HT). Let At be a finite set of atoms called the *(propositional) signature*. A *(propositional) formula* φ is defined using the grammar:

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

where p is an atom $p \in At$. We will use Greek letters φ, ψ, γ and their variants to stand for formulas. We define the derived operators $\neg\varphi \overset{\text{def}}{=} (\varphi \rightarrow \bot)$, $\top \overset{\text{def}}{=} \neg\bot$ and $\varphi \leftrightarrow \psi \overset{\text{def}}{=} (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$. In [1], we also included disjunction as an elementary

[1] For simplicity, we understand programs as the conjunction of their rules.
[2] As suggested, and partly conjectured, by the AIJ reviewers for [1].

connective, something usual in intuitionistic and intermediate logics. For the current work, however, we are interested in reducing the number of connectives in proofs, so we use the fact that disjunction in the logic HT can be defined [9] as follows:

$$\varphi \vee \psi \overset{\text{def}}{=} ((\varphi \to \psi) \to \psi) \wedge ((\psi \to \varphi) \to \varphi) \tag{1}$$

Given a formula φ, by $At(\varphi) \subseteq At$ we denote the set of atoms occurring in φ. A *literal* is an atom p or its negation $\neg p$. A *(logic) program* is a set of implications of the form $\alpha \to \beta$ where α is a conjunction of literals and β a disjunction of literals. A *theory* is a set of formulas. 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 satisfying $H \subseteq T \subseteq At$; 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:

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

It can be checked that the interpretation for disjunction when defined as (1) amounts to:

- $\langle H, T\rangle \models \varphi \vee \psi$ if $\langle H, T\rangle \models \varphi$ or $\langle H, T\rangle \models \psi$.

We proceed now to recall the definitions introduced in [1] that will be used for the main results.

Definition 1 *(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 $\mathcal{H} \neq \emptyset$. We write $\mathbf{H}_T$ to stand for the set of all possible T-supports.

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

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 $\mathcal{H} \supseteq \mathcal{H}' \neq [\ ]$.

As shown in [1, Proposition 4], the relation $\preceq_T$ constitutes 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.

Given a T-support $\mathcal{H}$, we define its complementary support $\overline{\mathcal{H}}$ as:

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

The relation between T-supports and formulas is given by the following definition.

Definition 3 *(T-**denotation**).* Let $T \subseteq At$. The T-*denotation* of a formula φ, written $[\![\varphi]\!]^T$, is a T-support recursively defined as follows:

$$\begin{aligned}
[\![\bot]\!]^T &\overset{\text{def}}{=} [\ ] \\
[\![p]\!]^T &\overset{\text{def}}{=} \{H \subseteq T \mid p \in H\} \\
[\![\varphi \wedge \psi]\!]^T &\overset{\text{def}}{=} [\![\varphi]\!]^T \cap [\![\psi]\!]^T \\
[\![\varphi \to \psi]\!]^T &\overset{\text{def}}{=} \begin{cases} [\ ] & \text{if } [\![\varphi]\!]^T \neq [\ ] \text{ and } [\![\psi]\!]^T = [\ ] \\ \overline{[\![\varphi]\!]^T} \cup [\![\psi]\!]^T & \text{otherwise} \end{cases}
\end{aligned}$$

Using this definition and Proposition 6 from [1], we obtain the following derived denotations for disjunction and negation:

$$[\![\varphi \vee \psi]\!]^T = [\![\varphi]\!]^T \cup [\![\psi]\!]^T \qquad [\![\neg\varphi]\!]^T = \begin{cases} [\ ] & \text{if } [\![\varphi]\!]^T \neq [\ ] \\ 2^T & \text{otherwise} \end{cases}$$

Propositional formulas (and logic programs) seen so far were extended in [1] to include a new connective '|', forming new expressions called *forks*. A *fork* F is defined by the grammar:

$$F \;::=\; \bot \;\big|\; p \;\big|\; (F \mid F) \;\big|\; F \wedge F \;\big|\; \varphi \vee \varphi \;\big|\; \varphi \to F$$

where φ is a propositional formula and $p \in At$ is an atom. We refer to the language formed by all forks for signature At as $\mathcal{L}_{At}$. Notice that the fork operator '|' cannot occur in the scope of negation, since $\neg F$ stands for $F \to \bot$ and implications do not allow '|' in the antecedent. For the same reason, the fork '|' cannot occur in a disjunction either, since (1) would require using that operator in the antecedent of an implication. In the current document, to make inductive proofs simpler, we introduce an alternative definition of $\mathcal{L}_{At}$ based on a partition of sublanguages $\mathcal{L}^i_{At}$ with respect to some degree $i \geq 0$ for connective nesting. In what follows, we will use the function $\delta(F)$ (the *degree* of fork F) to be defined as value i when $F \in \mathcal{L}^i_{At}$ has been already defined.

Definition 4 *(**Well formed fork**).* Given a set of propositional atoms At, we define the set of *well formed forks* for some degree $i \geq 0$, denoted as $\mathcal{L}^i_{At}$, inductively as follows:

$$\mathcal{L}^0_{At} \stackrel{\text{def}}{=} \text{the set of propositional formulas for } At$$

$$\begin{array}{rclll}
\mathcal{L}^{i+1}_{At} & \stackrel{\text{def}}{=} & \{ & (F_1 \mid \ldots \mid F_m) & \mid\; m > 1,\ \max\{\delta(F_1), \ldots, \delta(F_m)\} = i \quad \} \\
& \cup & \{ & (F_1 \wedge \cdots \wedge F_m) & \mid\; m > 1,\ \max\{\delta(F_1),\ \ldots, \delta(F_m)\} = i > 0 \quad \} \\
& \cup & \{ & (\varphi \to F) & \mid\; \delta(\varphi) = 0,\ \delta(F) = i > 0 \quad \}
\end{array}$$

The set of all well formed forks for At is defined as $\mathcal{L}_{At} \stackrel{\text{def}}{=} \bigcup_{i \geq 0} \mathcal{L}^i_{At}$.

Apart from partitioning the language by degrees, Definition 4 also introduces another minor variation with respect to the syntax in [1]: conjunction and '|' are defined here as m-ary operators, for an arbitrary $m > 1$, rather than as binary connectives. Given that these connectives are associative, this avoids their unnecessary nesting when repeated, producing a more economic and readable translation of forks into logic programs, as we will see later on. As an example to illustrate the definition of fork degree, the conjunction $p \wedge q$ has degree $\delta(p \wedge q) = 0$ because it is a propositional formula, but cannot be understood as a conjunction of forks $F \wedge G$ of some degree $i + 1$ because, as we see in Definition 4, this requires that either F or G (or both) have non-zero degrees. On the other hand, fork $(p \mid q)$ has a degree 1, since both p and q have degree 0 but the fork connective increases the degree by one. For a larger example, fork $s \to (((p \mid q) \mid r) \wedge (p \mid q))$ has a degree of 4 because the degree of an implication is the degree of its consequent plus one, and the latter is constructively explained below:

$$\underbrace{(\;\underbrace{(\;\overbrace{(p \mid q)}^{\max\{0,0\}+1=1} \;\mid r)}_{\max\{1,0\}+1=2} \;\wedge\; \underbrace{(p \mid q)}_{\max\{0,0\}+1=1}\;)}_{\max\{2,1\}+1=3}$$

Note that since ' | ' is associative (see Proposition 2 below), these forks can be rewritten as $s \to ((p \mid q \mid r) \wedge (p \mid q))$. This fork makes use of the m-ary operations of Definition 4 and it is strongly equivalent to the former. However, it has degree 3 rather than 4. We define the *size* of a fork F, written $|F|$, as the number of connectives and atom occurrences in F. For instance, $|\; s \to ((p \mid q \mid r) \wedge (p \mid q)) \;| = 11$.

The semantics of forks is defined in terms of sets of T-supports that we call T*-views*. 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 Δ so that:

$$^{\downarrow}\Delta \stackrel{\text{def}}{=} \bigcup_{\mathcal{H} \in \Delta} \downarrow\mathcal{H} \;=\; \{\, \mathcal{H}' \;\big|\; \mathcal{H}' \preceq \mathcal{H}, \mathcal{H} \in \Delta \,\} \setminus \{\,[\;]\,\}$$

Definition 5 *(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 Δ is a T-view and the $\preceq$-greatest T-support $[\;T\;]$ is included in Δ, then Δ is precisely $\downarrow[\;T\;]$. We proceed next to define the semantics of forks in terms of T-views.

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:

$$\begin{aligned}
\langle\!\langle\, F \,\rangle\!\rangle^T &\overset{\text{def}}{=} \downarrow [\![\, F \,]\!]^T \quad \text{if } \delta(F) = 0 \\
\langle\!\langle\, G_1 \wedge \ldots \wedge G_m \,\rangle\!\rangle^T &\overset{\text{def}}{=} {}^{\downarrow}\{\mathcal{H}_1 \cap \ldots \cap \mathcal{H}_m \mid \\
&\qquad\qquad \text{for each } \langle \mathcal{H}_1, \ldots, \mathcal{H}_m \rangle \in \langle\!\langle\, G_1 \,\rangle\!\rangle^T \times \cdots \times \langle\!\langle\, G_m \,\rangle\!\rangle^T \,\} \\
\langle\!\langle\, G_1 \mid \ldots \mid G_m \,\rangle\!\rangle^T &\overset{\text{def}}{=} \langle\!\langle\, G_1 \,\rangle\!\rangle^T \cup \ldots \cup \langle\!\langle\, G_m \,\rangle\!\rangle^T
\end{aligned}$$

$$\langle\!\langle\, \varphi \to G \,\rangle\!\rangle^T \overset{\text{def}}{=} \begin{cases} \{2^T\} & \text{if } [\![\, \varphi \,]\!]^T = [\;] \\ {}^{\downarrow}\{\, \overline{[\![\, \varphi \,]\!]^T} \cup \mathcal{H} \;\big|\; \mathcal{H} \in \langle\!\langle\, G \,\rangle\!\rangle^T \,\} & \text{otherwise} \end{cases}$$

In the last three cases, we assume $\delta(F) > 0$.

Finally, we reproduce the definition of the stable models of a fork from [1].

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

3. Invariance results for projective strong equivalence

In this section we revisit the definition of Projective Strong Equivalence (PSE) from [1] and provide several useful invariance results that will be used later on in our reduction to logic programs. As explained before, the main idea of PSE is that only a subset of atoms $V \subseteq At$ is considered public, whereas $At \setminus V$ are hidden. Given a set of sets of atoms $A \subseteq 2^{At}$, we denote its projection onto some vocabulary $V \subseteq At$ as $A_V \overset{\text{def}}{=} \{\, X \cap V \;\big|\; X \in A \,\}$.

Definition 8 ***(projective strong entailment/equivalence of forks)***. Let F and G be two forks and $V \subseteq At$ some vocabulary (set of atoms). We say that F *V-strongly entails* G, written $F \mathrel{|\!\!\sim}_V G$, if $SM_V[F \wedge L] \subseteq SM_V[G \wedge L]$ for any fork $L \in \mathcal{L}_V$. We further say that F and G are *V-strongly equivalent*, written $F \cong_V G$, if both $F \mathrel{|\!\!\sim}_V G$ and $G \mathrel{|\!\!\sim}_V F$, that is, $SM_V[F \wedge L] = SM_V[G \wedge L]$ for any fork $L \in \mathcal{L}_V$.

When $V \supseteq At(F) \cup At(G)$ we just remove the V and simply talk about (the non-projective versions of) *strong entailment* '$\mathrel{|\!\!\sim}$' and *strong equivalence* '$\cong$'.

A particular application of $\cong_V$ is the case where we consider F to be the "original" expression and G the result of some transformation $t(F)$ on F. We say that a transformation $t(F)$ is *strongly faithful* (adapted from [10]) with respect to F when $F \cong_V t(F)$ fixing the public vocabulary to $V = At(F)$.

The following result shows that $\mathrel{|\!\!\sim}$ and $\cong$ have a simple characterisation in terms of denotations.

Proposition 1 *(Proposition 11 in [1]). For any pair of forks F, G the following hold:*

(i) *$F \mathrel{|\!\!\sim} G$ iff for every set $T \subseteq At$, $\langle\!\langle\, F \,\rangle\!\rangle^T \subseteq \langle\!\langle\, G \,\rangle\!\rangle^T$,*
(ii) *$F \cong G$ iff for every set $T \subseteq At$, $\langle\!\langle\, F \,\rangle\!\rangle^T = \langle\!\langle\, G \,\rangle\!\rangle^T$.*

We begin introducing several useful equivalences among forks, proving that their versions for binary connectives '$\wedge$' and '$|$' in [1] also apply to the m-ary case.

Proposition 2. *Let $F_1, \ldots, F_m$ be arbitrary forks with $m > 2$. Then:*

$$\begin{aligned}
F_1 \mid \ldots \mid F_m \quad &\cong \quad F_1 \mid (F_2 \mid \ldots \mid F_m) \quad \cong \quad (F_1 \mid \ldots \mid F_{m-1}) \mid F_m \\
F_1 \wedge \ldots \wedge F_m \quad &\cong \quad F_1 \wedge (F_2 \wedge \ldots \wedge F_m) \quad \cong \quad (F_1 \wedge \ldots \wedge F_{m-1}) \wedge F_m
\end{aligned}$$

Proof. Let $T \subseteq At$. Then, by definition, we get:

$$\begin{aligned}
\langle\!\langle\, F_1 \mid \ldots \mid F_m \,\rangle\!\rangle^T = \bigcup_{i=1}^{m} \langle\!\langle\, F_i \,\rangle\!\rangle^T &= \langle\!\langle\, (F_1 \mid \ldots \mid F_{m-1}) \mid F_m \,\rangle\!\rangle^T \\
&= \langle\!\langle\, F_1 \mid (F_2 \mid \ldots \mid F_m) \,\rangle\!\rangle^T
\end{aligned}$$

For the case of conjunction, given $\mathcal{H}_i \in \langle\!\langle\, F_i \,\rangle\!\rangle^T$ for $i = 1, \ldots, m$, we know:

$$\mathcal{H}_1 \cap \mathcal{H}_2 \cap \ldots \cap \mathcal{H}_m = \mathcal{H}_1 \cap (\mathcal{H}_2 \cap \ldots \cap \mathcal{H}_m) = (\mathcal{H}_1 \cap \ldots \cap \mathcal{H}_{m-1}) \cap \mathcal{H}_m,$$

because set intersection is associative. Since $\mathcal{H}_2 \cap \ldots \cap \mathcal{H}_m \in \langle\!\langle\, F_2 \wedge \ldots \wedge F_m \,\rangle\!\rangle^T$ and $\mathcal{H}_1 \cap \ldots \cap \mathcal{H}_{m-1} \in \langle\!\langle\, F_1 \wedge \ldots \wedge F_{m-1} \,\rangle\!\rangle^T$ we obtain the result for m-ary conjunctions. □

Proposition 2 can be immediately applied to Proposition 12 in [1] to obtain the next useful equivalences

Corollary 1. *Let $F_1, \ldots, F_m$ and G be arbitrary forks and φ and ψ be formulas. Then:*

$$(F_1 \mid \ldots \mid F_m) \wedge G \cong (F_1 \wedge G) \mid \ldots \mid (F_m \wedge G) \tag{2}$$

$$\varphi \to (F_1 \mid \ldots \mid F_m) \cong (\varphi \to F_1) \mid \ldots \mid (\varphi \to F_m) \tag{3}$$

$$\varphi \to (F_1 \wedge \ldots \wedge F_m) \cong (\varphi \to F_1) \wedge \ldots \wedge (\varphi \to F_m) \tag{4}$$

$$\varphi \to (\psi \to F) \cong \varphi \wedge \psi \to F \tag{5}$$

$$\top \to F \cong F \tag{6}$$

The denotational characterisation of PSE relies on the following definition: we say that a T-support $\mathcal{H}$ is V*-unfeasible*[3] iff there is some $H \subset T$ in $\mathcal{H}$ satisfying $H \cap V = T \cap V$; we call it V*-feasible* otherwise.

Definition 9. 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:

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

In other words, we collect 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 $\langle\!\langle\, F \,\rangle\!\rangle^{T'}$. It has been proved in [1] that, for any $V \subseteq At$, the projected versions $\mathrel{|\!\sim}_V$ and $\cong_V$ have simple characterisations in terms of V-T-denotations:

Proposition 3 *(Theorem 2 in [1]). For any vocabulary $V \subseteq At$, forks F, G, the following hold:*

(i) *$F \mathrel{|\!\sim}_V G$ iff for every set $T \subseteq V$ of atoms, $\langle\!\langle\, F \,\rangle\!\rangle^T_V \subseteq \langle\!\langle\, G \,\rangle\!\rangle^T_V$, and*
(ii) *$F \cong_V G$ iff for every set $T \subseteq V$ of atoms, $\langle\!\langle\, F \,\rangle\!\rangle^T_V = \langle\!\langle\, G \,\rangle\!\rangle^T_V$.*

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 4 *(Proposition 13 in [1]). For any vocabulary $V \subseteq At$, fork F with $At(F) \subseteq V$ and set $T \subseteq V$ of atoms, $\langle\!\langle\, F \,\rangle\!\rangle^T_V = \langle\!\langle\, F \,\rangle\!\rangle^T$.*

The following theorem from [1] guarantees that V-strong entailment (and so, V-strong equivalence too) is unaffected by any atom a not occurring in F or G.

Theorem 1 *(**Free Atom Invariance**, Theorem 3 in [1]). Let F and G be two forks and let At be a signature such that $At \supset At(F) \cup At(G)$ and $a \in At \setminus (At(F) \cup At(G))$, for some atom a. For any $V \subseteq At$ we have: $F \mathrel{|\!\sim}_V G$ for signature At iff $F \mathrel{|\!\sim}_{V'} G$ for signature $At' = At \setminus \{a\}$ and $V' = V \setminus \{a\}$.*

We state next another pair of useful invariance properties about projective strong equivalence: Reduced Vocabulary (Proposition 5) and Public Context (Proposition 6). The first proposition guarantees that $F \mathrel{|\!\sim}_V G$ is preserved if we replace V by any smaller vocabulary $V' \subseteq V$. To prove that result, we rely on the following lemma.

Lemma 1. *Let $A, B \subseteq 2^{At}$ and let $V' \subseteq V$ and, $A_V \subseteq B_V$. Then, $A_{V'} \subseteq B_{V'}$.*

Proof. Suppose $T \in A_{V'}$, that is, $T = T_1 \cap V'$ for some $T_1 \in A$. Then, $(T_1 \cap V) \in A_V \subseteq B_V$. As $(T_1 \cap V) \in B_V$, there exists $T_2 \in B$ such that $(T_2 \cap V) = (T_1 \cap V)$. Given $V' \subseteq V$, we conclude $(T_2 \cap V') = (T_1 \cap V') = T$. Finally, $T_2 \in B$ implies $T = (T_2 \cap V') \in B_{V'}$. □

Proposition 5 *(**Reduced Vocabulary Invariance**). Let F, G be a pair of forks satisfying $F \mathrel{|\!\sim}_V G$ and let $V' \subseteq V$. Then $F \mathrel{|\!\sim}_{V'} G$.*

[3] This notion is analogous to condition *ii)* in the definition of V-SE-models that characterises relativised strong equivalence [11].

Proof. If $F \mathrel{|\!\sim}_V G$ then $SM_V[F \wedge L] \subseteq SM_V[G \wedge L]$ for all $L \in \mathcal{L}_V$. From $V' \subseteq V$ and Lemma 1 we conclude $SM_{V'}[F \wedge L] \subseteq SM_{V'}[G \wedge L]$ for all $L \in \mathcal{L}_V \supseteq \mathcal{L}_{V'}$, and so $F \mathrel{|\!\sim}_{V'} G$. □

Strong (Addition) Invariance corresponds to a property of forgetting operators first identified by Wong in [12] and later dubbed with that name in [13]. In the case of forgetting, if this property holds, it means that if we add any program fragment without the forgotten atoms, we can do it before or after forgetting and the results in both cases are strongly equivalent. In the case of $F \mathrel{|\!\sim}_V G$, a somehow similar property, we call *Public Context Invariance* (PCI), determines that we can always add any context C over vocabulary V to both F and G and the strong entailment relation is preserved.

Proposition 6 (***Public Context Invariance, PCI***). *Let F, G be a pair of forks satisfying $F \mathrel{|\!\sim}_V G$ and let $C \in \mathcal{L}_V$. Then, $F \wedge C \mathrel{|\!\sim}_V G \wedge C$.*

Proof. We prove that $F \wedge C \not\mathrel{|\!\sim}_V G \wedge C$ implies $F \not\mathrel{|\!\sim}_V G$. Assume $F \wedge C \not\mathrel{|\!\sim}_V G \wedge C$. Then, there is some fork $L \in \mathcal{L}_V$ s.t. $SM_V[(F \wedge C) \wedge L] \not\subseteq SM_V[(G \wedge C) \wedge L]$. Now, the fork $L' = (C \wedge L)$ is also in $\mathcal{L}_V$ and, since conjunction is associative, we get $SM_V[F \wedge L'] \not\subseteq SM_V[G \wedge L']$. Hence, $F \not\mathrel{|\!\sim}_V G$. □

To conclude this section, we further generalise PCI by showing that it still holds when C contains other atoms not in V, but does not use the "hidden" atoms in F and G. In other words, any atom in C occurring in F or G must belong to the public vocabulary V.

Theorem 2 (***Hidden Atoms Invariance***). *Let F, G be two forks such that $F \cong_V G$. Then $F \wedge C \cong_V G \wedge C$ for any fork C satisfying $At(C) \cap (At(F) \cup At(G)) \subseteq V$.*

Proof. Atoms in $At(C) \setminus V$ do not belong to $At(F) \cup At(G)$, so they are free atoms with respect to $F \cong_V G$. We can incrementally apply free atom invariance (Theorem 1) on atoms in $At(C) \setminus V$, eventually adding all of them to V to conclude $F \cong_{V'} G$ for $V' = V \cup At(C)$. Now, since $At(C) \subseteq V'$, by Proposition 6, $F \cong_{V'} G$ implies $F \wedge C \cong_{V'} G \wedge C$. Finally, by Proposition 5 and $V \subseteq V'$ we conclude $F \wedge C \cong_V G \wedge C$. □

To illustrate the utility of these results, take program P_m and assume that the public vocabulary is $V = \{a, b\}$, so its local atoms are $\{ma, mb\}$. As we explained in the introduction[4] both $P_m \cong_V (a \mid b)$ and $P_f \cong_V (a \mid b)$. Suppose we take $F = P_m$, $G = (a \mid b)$ and $C = P_f$. Note that $C = P_f$ has atoms $\{fa, fb\}$ not in V, but these atoms do not occur in F or G. Therefore, we can apply Hidden Atoms Invariance to conclude $F \wedge C \cong_V G \wedge C$, that is,

$$P_m \wedge P_f \cong_V (a \mid b) \wedge P_f \tag{7}$$

But now, we can replace P_f on the right hand side above by another fork as follows. Take $F = P_f$, $G = (a \mid b)$ and $C = (a \mid b)$. In this case all atoms in C are public and, by PCI (Proposition 6), we conclude $F \wedge C \cong_V G \wedge C$, that is:

$$P_f \wedge (a \mid b) \cong_V (a \mid b) \wedge (a \mid b) \tag{8}$$

Since $\wedge$ is symmetric and $\cong_V$ is transitive, from (7) and (8) we can finally conclude $P_m \wedge P_f \cong_V (a \mid b) \wedge (a \mid b)$.

4. Reduction to propositional formulas and logic programs

We are now ready to introduce the new polynomial reduction of any fork F into a propositional formula $pf(F)$ that may introduce auxiliary atoms, but is $At(F)$-strongly equivalent to F. In fact, the propositional formula we obtain $\varphi = pf(F)$ is not necessarily in the form of a logic program, but it can be further reduced to that form using the polynomial method in [14], that introduces again auxiliary variables, being strongly faithful (i.e. keeping PSE for the original alphabet). To define $pf(F)$, we introduce first a recursive transformation $im(\cdot)$ that exclusively operates on forks that have the form of an implication $\varphi \to F$.

[4] See Example 8 in [1] for more details.

Definition 10. Given a fork of the form $\varphi \to F$ by $im(\varphi \to F)$ we denote the following recursive rewriting:

$$\begin{array}{rcl} im(\varphi \to F) & \stackrel{\text{def}}{=} & \varphi \to F \quad \text{if } F \text{ is a propositional formula} \\ im(\varphi \to (F_1 \mid \ldots \mid F_m)) & \stackrel{\text{def}}{=} & \big(\varphi \to (a_1 \vee \ldots \vee a_m)\big) \wedge \bigwedge_{i=1}^{m} im(a_i \to F_i) \\ & & \text{where each } a_i \text{ is a fresh atom} \\ im(\varphi \to (F_1 \wedge \ldots \wedge F_m)) & \stackrel{\text{def}}{=} & (\varphi \to a) \wedge \bigwedge_{i=1}^{m} im(a \to F_i) \\ & & \text{if } F_1 \wedge \ldots \wedge F_m \text{ is not a formula and} \\ & & a \text{ is a fresh atom} \\ im(\varphi \to (\psi \to F)) & \stackrel{\text{def}}{=} & im(\varphi \wedge \psi \to F) \quad \text{if } F \text{ is not a formula} \qquad \square \end{array}$$

As we will prove later, it is not difficult to see that $im(\varphi \to F)$ is indeed a propositional formula, but its application is limited to forks of the form $\varphi \to F$. Fortunately, if the original fork F is not in that form, we can always replace it by $\top \to F$ and then apply $im(\top \to F)$. The general transformation $pf(F)$ is then defined as follows.

Definition 11 *(**The** $pf(\cdot)$ **reduction**).* For any fork F we define:

$$\begin{array}{lcl} pf(F) & \stackrel{\text{def}}{=} & F \quad \text{if } F \text{ is a propositional formula} \\ pf(F) & \stackrel{\text{def}}{=} & im(\varphi \to G) \quad \text{if } F = \varphi \to G \text{ and } \delta(G) > 0 \\ pf(F) & \stackrel{\text{def}}{=} & im(\top \to F) \quad \text{otherwise} \end{array}$$

Now, the properties of this reduction are captured by the main theorem below, whose detailed proof will be provided in the rest of the section.

Main Theorem. *For any fork F, the following statements hold:*

1. *$pf(F)$ is a propositional formula,*
2. *$pf(F) \cong_{At(F)} F$,*
3. *$|pf(F)| \leq 3\,|F|^2$, and*
4. *$pf(\cdot)$ can be computed in polynomial time.* $\square$

To illustrate the application of this transformation, let F_1 be the fork:

$$(p \mid \neg r) \wedge \Big(\neg p \to \big(\,(q \to (p \mid r)) \wedge (\neg q \to (r \mid s))\big)\Big)$$

We start with $pf(F_1) = im(\top \to F_1)$. Then, we introduce a_0 to replace the (conjunctive) F_1 in the consequent, leading to $(\top \to a_0) \wedge im(a_0 \to (p \mid \neg r)) \wedge im(a_0 \to (\neg p \to G))$ where we write G to abbreviate the consequent of the second conjunct in F_1. The application $im(a_0 \to (p \mid \neg r))$ introduces two new auxiliary atoms leading to $(a_0 \to a_1 \vee a_2) \wedge (a_1 \to p) \wedge (a_2 \to \neg r)$. On the other hand, $im(a_0 \to (\neg p \to G)) = im(a_0 \wedge \neg p \to G)$. We proceed similarly with G and eventually obtain $pf(F_1)$ as the conjunction of:

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl} \top & \to & a_0 & a_0 \wedge \neg p & \to & a_3 & a_3 \wedge \neg q & \to & a_6 \vee a_7 \\ a_0 & \to & a_1 \vee a_2 & a_3 \wedge q & \to & a_4 \vee a_5 & a_6 & \to & r \\ a_1 & \to & p & a_4 & \to & p & a_7 & \to & s \\ a_2 & \to & \neg r & a_5 & \to & r & & & \end{array}$$

that, in this case, it already has the form of a logic program, not requiring the further reduction in [14]. The newly introduced atoms $a_0, \ldots, a_7$ are auxiliary. The Main Theorem guarantees that the resulting program is V-strongly equivalent to F_1, where $V = At(F_1) = \{p, q, r, s\}$. Moreover, by Theorem 1 (Free Atom Invariance), we know that V'-strong equivalence still holds for any extended public vocabulary $V' \supseteq V$ that does not contain the hidden atoms $a_0, \ldots, a_7$. Finally, the Main Theorem also states that, in the worst case, the size of the reduction $pf(F)$ remains quadratic.

Before we proceed to prove the Main Theorem, we start identifying a particular kind of T-support whose singularity will be exploited later on.

Definition 12 *(V**-respectful support**).* Let $T, V \subseteq At$ be two sets of atoms. We say that a T-support $\mathcal{H}$ is *V-respectful*, if any $H, H' \subseteq T$ such that $H \cap V = H' \cap V$ satisfies $H \in \mathcal{H}$ iff $H' \in \mathcal{H}$.

To start with the proof, we provide a pair of basic transformations, γ and λ, that allow removing fork connectives in the consequent of an implication, but at the cost of introducing auxiliary atoms. Some parts of the proof for their PSE make use of Lemmata 5 and 6 from [1].

Lemma 2. *Let $F = \varphi \to (F_1 \mid \cdots \mid F_n)$ be a fork, $V = At(F)$ and let*

$$\gamma(F) \overset{def}{=} (a_1 \vee \cdots \vee a_n) \wedge (a_1 \wedge \varphi \to F_1) \wedge \cdots \wedge (a_n \wedge \varphi \to F_n)$$
$$\lambda(F) \overset{def}{=} [\varphi \to (a_1 \vee \cdots \vee a_n)] \wedge (a_1 \to F_1) \wedge \cdots \wedge (a_n \to F_n)$$

where $a_i \notin V$ for all $1 \le i \le n$. Then, $F \cong_V \gamma(F) \cong_V \lambda(F)$.

Proof. Taking into account Proposition 3, we have to prove:

$$\langle\!\langle F \rangle\!\rangle_V^T = \langle\!\langle \gamma(F) \rangle\!\rangle_V^T = \langle\!\langle \lambda(F) \rangle\!\rangle_V^T$$

for any $T \subseteq V$. On the other hand, Proposition 4 and (3) guarantee:

$$\langle\!\langle F \rangle\!\rangle_V^T = \langle\!\langle F \rangle\!\rangle^T = \langle\!\langle \varphi \to (F_1 \mid \cdots \mid F_n) \rangle\!\rangle^T = \bigcup_{i=1}^{n} \langle\!\langle \varphi \to F_i \rangle\!\rangle^T$$

So, in the end, what we have to prove is:

$$\bigcup_{i=1}^{n} \langle\!\langle \varphi \to F_i \rangle\!\rangle^T = \langle\!\langle \gamma(F) \rangle\!\rangle_V^T = \langle\!\langle \lambda(F) \rangle\!\rangle_V^T$$

We decompose this equality into a chain of three inclusion relations.

First inclusion: $\bigcup_{i=1}^{n} \langle\!\langle \varphi \to F_i \rangle\!\rangle^T \subseteq \langle\!\langle \gamma(F) \rangle\!\rangle_V^T$.
Take $1 \le i \le n$ and $\mathcal{H} \in \langle\!\langle \varphi \to F_i \rangle\!\rangle^T$. We distinguish two cases depending on whether $T \models \varphi$ or not. Suppose first that $T \not\models \varphi$. Then $\mathcal{H} = 2^T$ and, because of Lemma 5 from [1], if $S = T \cup \{a_i\}$, then $S \not\models \varphi$. Notice that $\{2^S\} = \langle\!\langle \varphi \wedge a_j \to F_j \rangle\!\rangle^S$ for any $1 \le j \le n$. Then:

$$\begin{aligned} 2^T &= [\![a_1 \vee \cdots \vee a_n]\!]_V^S \\ &= \left([\![a_1 \vee \cdots \vee a_n]\!]^S \cap 2^S \cap \cdots \cap 2^S \right)_V \in \langle\!\langle \gamma(F) \rangle\!\rangle_V^T. \end{aligned}$$

In the second case, if $T \models \varphi$, $\mathcal{H} \preceq \overline{[\![\varphi]\!]^T} \cup \mathcal{H}'$ with $\mathcal{H}' \in \langle\!\langle F_i \rangle\!\rangle^T$ maximal. We define $\mathcal{H}' \cup \{a_i\} := \{H \cup \{a_i\}\ ;\ H \in \mathcal{H}'\}$ which is an S-support if $S = T \cup \{a_i\}$. Notice that

$$\begin{aligned} & [\![a_1 \vee \cdots \vee a_n]\!]^S \cap [\overline{[\![\varphi \wedge a_i]\!]^S} \cup (\mathcal{H}' \cup \{a_i\})] \\ = \ & \{X \subseteq S\ ;\ a_i \in X \text{ and } \langle X, S\rangle \not\models \varphi\} \\ & \cup\ \{X \subseteq S\ ;\ a_i \in X \text{ and } X \setminus \{a_i\} \in \mathcal{H}'\} \end{aligned}$$

but this implies:

$$\overline{[\![\varphi]\!]^T} \cup \mathcal{H}' = \left([\![a_1 \vee \cdots \vee a_n]\!]^S \cap [\overline{[\![\varphi \wedge a_i]\!]^S} \cup (\mathcal{H}' \cup \{a_i\})] \right)_V$$

and the latter belongs to $\langle\!\langle \gamma(F) \rangle\!\rangle_V^T$ since $\langle\!\langle \varphi \wedge a_j \to F_j \rangle\!\rangle^S = \{2^S\}$ for any $j \neq i$, $[\![a_1 \vee \cdots \vee a_n]\!]^S$ is V-feasible[5] and $\mathcal{H} \cup \{a_i\} \in \langle\!\langle F_i \rangle\!\rangle^S$ (by Lemma 20 from [1]). The fact that $\mathcal{H}'$ is V-respectful (because of Lemma 6 from [1]) implies that, for any $X \subseteq S$, $X \in \mathcal{H}'$ if, and only if, $X \cup \{a_i\} \in \mathcal{H}'$ since $X \cap V = (X \cup \{a_i\}) \cap V$.

Second inclusion: $\langle\!\langle \gamma(F) \rangle\!\rangle_V^T \subseteq \langle\!\langle \lambda(F) \rangle\!\rangle_V^T$.
For this inclusion, if we suppose that $\mathcal{H}_V \in \langle\!\langle \gamma(F) \rangle\!\rangle_V^T$ for some $\mathcal{H} \in \langle\!\langle \gamma(F) \rangle\!\rangle^S$ such that $S \cap V = T$ and with $\mathcal{H}$ being V-feasible, we can suppose that $S \neq T$ since $\langle\!\langle \gamma(F) \rangle\!\rangle^T = \emptyset$. Moreover, if $T \cup \{a_i\} \subset S$, any $\mathcal{H} \in \langle\!\langle \gamma(F) \rangle\!\rangle^S$ is going to be V-unfeasible. For any such $\mathcal{H}$, we know that there exist $\mathcal{H}_k \in \langle\!\langle \varphi \wedge a_k \to F_k \rangle\!\rangle^S$ (with $\underline{1 \le k \le n}$) and $\mathcal{H}_{n+1} \in \langle\!\langle a_1 \vee \cdots \vee a_n \rangle\!\rangle^S$ such that $\mathcal{H} \preceq \bigcap_{k=1}^{n} \mathcal{H}_k \cap \mathcal{H}_{n+1}$. We are going to use afterwards the fact that $\mathcal{H}_i \preceq \overline{[\![\varphi \wedge a_i]\!]^S} \cup \mathcal{H}'_i$ for some $\mathcal{H}'_i$ maximal in $\langle\!\langle F_i \rangle\!\rangle^S$. Since $\mathcal{H}'_i$ is V-respectful (by applying Lemma 6 from [1]) and $S \in \mathcal{H}'_i$, we have $T \cup \{a_i\} \in \mathcal{H}'_i \subseteq \mathcal{H}_i$.
If $T \not\models \varphi$, then $\mathcal{H}_k = 2^S$ for any $1 \le k \le n$ and $T \cup \{a_i\} \in [\![a_1 \vee \cdots \vee a_n]\!]^S \subseteq \mathcal{H}_{n+1}$ that implies

$$T \cup \{a_i\} \in \mathcal{H}_{n+1} \cap \bigcap_{k=1}^{n} \mathcal{H}_k \subseteq \mathcal{H}$$

so $\mathcal{H}$ is V- unfeasible. On the other hand, if $T \models \varphi$, then $T \cup \{a_i\} \in \mathcal{H}'_i \subseteq \mathcal{H}_i$ and $T \cup \{a_i\} \in \overline{[\![\varphi \wedge a_k]\!]^S} \subseteq \mathcal{H}_k$. If $k \neq i$ then

[5] By Lemma 10 from [1], if $\mathcal{H}$ is V-feasible, then $\mathcal{H} \cap \mathcal{H}'$ is also V-feasible, for any $\mathcal{H}'$.

$$T \cup \{a_i\} \in \mathcal{H}_{n+1} \cap \bigcap_{k=1}^{n} \mathcal{H}_k \subseteq \mathcal{H}$$

so $\mathcal{H}$ is again V-unfeasible. Now take $\mathcal{H}_V \in \langle\!\langle\, \gamma(F) \,\rangle\!\rangle_V^T$ with $\mathcal{H} \in \langle\!\langle\, \gamma(F) \,\rangle\!\rangle^S$ being $S = T \cup \{a_i\}$, for some $1 \leq i \leq n$ and such that $\mathcal{H}$ is V-feasible. As we said above, there exist $\mathcal{H}_k \in \langle\!\langle\, \varphi \wedge a_k \rightarrow F_k \,\rangle\!\rangle^S$ (with $1 \leq k \leq n$) and $\mathcal{H}_{n+1} \in \langle\!\langle\, a_1 \vee \cdots \vee a_n \,\rangle\!\rangle^S$ such that $\mathcal{H} \preceq \bigcap_{k=1}^{n} \mathcal{H}_k \cap \mathcal{H}_{n+1}$. We can actually divide the proof in two cases, depending on whether $T \models \varphi$ or not. Suppose first that $T \not\models \varphi$. Then, we obtain $\mathcal{H}_k = 2^S$ for any $1 \leq k \leq n$, so:

$$\begin{aligned}\mathcal{H}_V &\preceq \left(\mathcal{H}_{n+1} \cap \bigcap_{k=1}^{n} \mathcal{H}_k\right)_V \\ &\preceq [\![\, a_1 \vee \cdots \vee a_n \,]\!]_V^S = 2^T \in \langle\!\langle\, \lambda(F) \,\rangle\!\rangle^T \subseteq \langle\!\langle\, \lambda(F) \,\rangle\!\rangle_V^T,\end{aligned}$$

because $\langle\!\langle\, \varphi \rightarrow (a_1 \vee \cdots \vee a_n) \,\rangle\!\rangle^T = \{2^T\} = \langle\!\langle\, a_i \rightarrow F_i \,\rangle\!\rangle^T$ for any $i = 1, \ldots, n$. Now, for the second case, if $T \models \varphi$, then $\mathcal{H}_k = 2^S$ for any $k \neq i$ and:

$$\begin{aligned}\mathcal{H}_V &\preceq \left(\mathcal{H}_{n+1} \cap \bigcap_{k=1}^{n} \mathcal{H}_k\right)_V \\ &= (\mathcal{H}_{n+1} \cap \mathcal{H}_i)_V \preceq \left(\mathcal{H}_{n+1} \cap (\overline{[\![\, \varphi \wedge a_i \,]\!]^S} \cup \mathcal{H}_i')\right)_V\end{aligned}$$

with $\mathcal{H}_i' \in \langle\!\langle\, F_i \,\rangle\!\rangle^S$. Then:

$$\begin{aligned}\mathcal{H}_V &\preceq \left([\![\, a_1 \vee \cdots \vee a_n \,]\!]^S \cap (\overline{[\![\, \varphi \wedge a_i \,]\!]^S} \cup \mathcal{H}_i')\right)_V \\ &= \left(\{\, X \cup \{a_i\} \mid X \in \overline{[\![\, \varphi \,]\!]^T} \text{ or } X \in \mathcal{H}_i' \,\}\right)_V \\ &= \overline{[\![\, \varphi \,]\!]^T} \cup (\mathcal{H}_i')_V\end{aligned}$$

Now, take into account that: $\overline{[\![\, \varphi \,]\!]^T} \cup (\mathcal{H}_i')_V$ is equal to:

$$\left[\left(\overline{[\![\, \varphi \,]\!]^S} \cup [\![\, a_1 \vee \cdots \vee a_n \,]\!]^S\right) \cap \left(\overline{[\![\, a_i \,]\!]^S} \cup \mathcal{H}_i'\right)\right]_V \in \langle\!\langle\, \lambda(F) \,\rangle\!\rangle_V^T,$$

because:

$$\begin{aligned}&\left[\left(\overline{[\![\, \varphi \,]\!]^S} \cup [\![\, a_1 \vee \cdots \vee a_n \,]\!]^S\right) \cap \left(\overline{[\![\, a_i \,]\!]^S} \cup \mathcal{H}_i'\right)\right] \\ &= \{\, X \subseteq T \mid \langle X, T \rangle \not\models \varphi \,\} \cup \{\, X \in \mathcal{H}_i' \mid \langle X, S \rangle \not\models \varphi \text{ or } a_i \in X \,\}\end{aligned}$$

Third inclusion: $\langle\!\langle\, \lambda(F) \,\rangle\!\rangle_V^T \subseteq \bigcup_{i=1}^{n} \langle\!\langle\, \varphi \rightarrow F_i \,\rangle\!\rangle^T$
In this case, if we suppose that $\mathcal{H}_V \in \langle\!\langle\, \lambda(F) \,\rangle\!\rangle_V^T$ for some $\mathcal{H} \neq [\ ] \in \langle\!\langle\, \lambda(F) \,\rangle\!\rangle^S$ such that $S \cap V = T$ and with $\mathcal{H}$ being V-feasible, we can prove, in a similar way as we have done for $\langle\!\langle\, \gamma(F) \,\rangle\!\rangle_V^T$, that $S = T$ and $T \not\models \varphi$ (since $\langle\!\langle\, \lambda(F) \,\rangle\!\rangle^T = \emptyset$ if $T \models \varphi$) or $S = T \cup \{a_i\}$ for some a_i.

- If $S = T$ and $T \not\models \varphi$, then $\mathcal{H} = 2^T$ and $\mathcal{H}_V = 2^T \in \bigcup_{i=1}^{n} \langle\!\langle\, \varphi \rightarrow F_i \,\rangle\!\rangle^T$.
- When $S = T \cup \{a_i\}$ and $T \not\models \varphi$, we can say that there exists $\mathcal{H}_i \in \langle\!\langle\, F_i \,\rangle\!\rangle^S$ such that $\mathcal{H} \preceq 2^S \cap \left(\overline{[\![\, a_i \,]\!]^S} \cup \mathcal{H}_i\right) \preceq \overline{[\![\, a_i \,]\!]^S}$. So $\mathcal{H}_V \preceq (\overline{[\![\, a_i \,]\!]^S})_V = 2^T \in \bigcup_{i=1}^{n} \langle\!\langle\, \varphi \rightarrow F_i \,\rangle\!\rangle^T$.
- If $S = T \cup \{a_i\}$ and $T \models \varphi$, then, there exists $\mathcal{H}_i \in \langle\!\langle\, F_i \,\rangle\!\rangle^S$ maximal such that:

  $$\begin{aligned}\mathcal{H} &\preceq \left(\overline{[\![\, \varphi \,]\!]^S} \cup [\![\, a_1 \vee \cdots \vee a_n \,]\!]^S\right) \cap \left(\overline{[\![\, a_i \,]\!]^S} \cup \mathcal{H}_i\right) \\ &= \{\, X \subseteq T \mid \langle X, T \rangle \not\models \varphi \,\} \cup \{\, X \in \mathcal{H}_i \mid a_i \in X \text{ or } \langle X, S \rangle \not\models \varphi \,\}\end{aligned}$$

  Finally, we obtain:

  $$\begin{aligned}\mathcal{H}_V &\preceq \left[\left(\overline{[\![\, \varphi \,]\!]^S} \cup [\![\, a_1 \vee \cdots \vee a_n \,]\!]^S\right) \cap \left(\overline{[\![\, a_i \,]\!]^S} \cup \mathcal{H}_i\right)\right]_V \\ &= \overline{[\![\, \varphi \,]\!]^T} \cup (\mathcal{H}_i)_V \in \langle\!\langle\, \varphi \rightarrow F_i \,\rangle\!\rangle^T. \quad \square\end{aligned}$$

Lemma 3. *Let F be a fork and φ a formula such that $At(F) \cup At(\varphi) \subseteq V \subseteq At$. Then, for any $a \notin V$, we get:*

$$(\varphi \rightarrow F) \cong_V (\varphi \rightarrow a) \wedge (a \rightarrow F).$$

Proof. For any $T \subseteq V$, we have to prove:

$$\langle\!\langle \varphi \to F \rangle\!\rangle^T = \langle\!\langle \varphi \to F \rangle\!\rangle^T_V = \langle\!\langle (\varphi \to a) \wedge (a \to F) \rangle\!\rangle^T_V$$

1. First of all, if we take $\mathcal{H} \in \langle\!\langle (\varphi \to a) \wedge (a \to F) \rangle\!\rangle^S$ with $S \cap V = T$ and $\mathcal{H}$ being V-feasible, we can show that $S = T \cup \{a\}$.
   - When $T \not\models \varphi$, then $\langle\!\langle \varphi \to a \rangle\!\rangle^S = \{2^S\}$ and there exists $\mathcal{H}' \in \langle\!\langle F \rangle\!\rangle^S$ maximal (and then V-respectful) such that $\mathcal{H} \preceq \overline{[\![a]\!]^S} \cup \mathcal{H}'$. Since $T \cup \{a\} \in \mathcal{H}' \subseteq \mathcal{H}$, we deduce that $S = T \cup \{a\}$.
   - Suppose that $T \models \varphi$ and $a \notin S$. Then $\langle\!\langle a \to F \rangle\!\rangle^S = \{2^S\}$ and $\mathcal{H} \preceq \overline{[\![\varphi]\!]^S} \cup [\![a]\!]^S$, so we can say that $T \cup \{a\} \in [\![a]\!]^S \subseteq \mathcal{H}$ and $S = T \cup \{a\}$.
   - Finally, suppose that $T \models \varphi$ and $a \in S$. We know that there exists $\mathcal{H}' \in \langle\!\langle F \rangle\!\rangle^S$ maximal such that:

     $$\mathcal{H} \preceq (\overline{[\![\varphi]\!]^S} \cup [\![a]\!]^S) \cap (\overline{[\![a]\!]^S} \cup \mathcal{H}')$$

     which implies that $T \cup \{a\} \in \mathcal{H}$ and $S = T \cup \{a\}$.
2. Suppose that $T \not\models \varphi$. In this case

   $$\langle\!\langle \varphi \to F \rangle\!\rangle^T \subseteq \langle\!\langle (\varphi \to a) \wedge (a \to F) \rangle\!\rangle^T_V$$

   since $\langle\!\langle \varphi \to F \rangle\!\rangle^T = \{2^T\} = \langle\!\langle \varphi \to a \rangle\!\rangle^T = \langle\!\langle a \to F \rangle\!\rangle^T$.
   For the other inclusion, if $[\ ] \neq \mathcal{H} \in \langle\!\langle (\varphi \to a) \wedge (a \to F) \rangle\!\rangle^S$, with $S \cap V = T$ and $\mathcal{H}$ being V-feasible, then we already know that $S = T \cup \{a\}$ and $\mathcal{H} \preceq \overline{[\![a]\!]^S} \cup \mathcal{H}'$ with $\mathcal{H}' \in \langle\!\langle F \rangle\!\rangle^S$. Then, $\mathcal{H} \preceq \overline{[\![a]\!]^S}$ which implies that $\mathcal{H}_V \preceq (\overline{[\![a]\!]^S})_V = 2^T \in \langle\!\langle \varphi \to F \rangle\!\rangle^T$.
3. Now, let's assume that $T \models \varphi$.
   - Take $\overline{[\![\varphi]\!]^T} \cup \mathcal{H} \in \langle\!\langle \varphi \to F \rangle\!\rangle^T$ for some $\mathcal{H} \in \langle\!\langle F \rangle\!\rangle^T$ maximal. If $S = T \cup \{a\}$, the support $\mathcal{H} \cup \{a\} = \{\ H \cup \{a\} \mid H \in \mathcal{H}\ \} \in \langle\!\langle F \rangle\!\rangle^S$ (from Lemma 20 in [1]) and:

     $$[\![\varphi \to a]\!]^S \cap (\overline{[\![a]\!]^S} \cup \mathcal{H} \cup \{a\}) = (\overline{[\![\varphi]\!]^T} \cup (\mathcal{H} \cup \{a\})$$

     which implies that

     $$\overline{[\![\varphi]\!]^T} \cup \mathcal{H} = [[\![\varphi \to a]\!]^S \cap (\overline{[\![a]\!]^S} \cup \mathcal{H} \cup \{a\})]_V$$

     so $\langle\!\langle \varphi \to F \rangle\!\rangle^T \subseteq \langle\!\langle (\varphi \to a) \wedge (a \to F) \rangle\!\rangle^T_V$.
   - Now suppose that $\mathcal{H} \in \langle\!\langle (\varphi \to a) \wedge (a \to F) \rangle\!\rangle^S$, with $S \cap V = T$ and $\mathcal{H}$ being V-feasible. We have already proved that $S = T \cup \{a\}$. Take $\mathcal{H}' \in \langle\!\langle F \rangle\!\rangle^S$ such that:

     $$\begin{aligned} \mathcal{H} &\preceq (\overline{[\![\varphi]\!]^S} \cup [\![a]\!]^S) \cap (\overline{[\![a]\!]^S} \cup \mathcal{H}') \\ &= (\overline{[\![\varphi]\!]^S} \cap \overline{[\![a]\!]^S}) \cup (\overline{[\![\varphi]\!]^S} \cap \mathcal{H}') \cup ([\![a]\!]^S \cap \mathcal{H}' \cap [\![\varphi]\!]^S) \\ &= \overline{[\![\varphi]\!]^T} \cup (\overline{[\![\varphi]\!]^S} \cap \mathcal{H}') \cup ([\![a]\!]^S \cap \mathcal{H}' \cap [\![\varphi]\!]^S) \end{aligned}$$

     Finally: $\mathcal{H}_V \preceq \overline{[\![\varphi]\!]^T} \cup \mathcal{H}'_V \in \langle\!\langle \varphi \to F \rangle\!\rangle^T$. $\square$

Lemmata 2 and 3 allow us to prove a similar result to the Main Theorem, but for the transformation $im(\varphi \to F)$ that only applies to implications.

Theorem 3. *For any fork of the form $\varphi \to F$, the following statements hold:*

1. *$im(\varphi \to F)$ is a propositional formula,*
2. *$(\varphi \to F) \cong_{At(\varphi \to F)} im(\varphi \to F)$,*
3. *$|im(\varphi \to F)| \leq |\varphi \to F|^2$, and*
4. *$im(\varphi \to F)$ can be computed in polynomial time.*

Proof. We proceed by induction on the degree of $\varphi \to F$. If $\varphi \to F$ is a propositional formula, we have nothing to prove since $im(\varphi \to F) = \varphi \to F$. Now, suppose that $\delta(\varphi \to F) > 0$.

- If $F = (\varphi \to F_1 \mid \ldots \mid F_m)$, then we get $F \cong (\varphi \to F_1) \mid \ldots \mid (\varphi \to F_m)$ from (3) and, from Lemma 2, we further obtain:

  $$F \cong_{At(F)} (\varphi \to (a_1 \vee \ldots \vee a_m)) \wedge \bigwedge_{i=1}^{m} (a_i \to F_i)$$

  Moreover,

$$\delta(a_i \to F_i) = 1 + \delta(F_i) \quad < \quad \delta(F) = 2 + \max\{\delta(F_i) \mid 1 \leq i \leq m\}$$

and, by the induction hypothesis, we get $(a_i \to F_i) \cong_{At(F_i)} im(a_i \to F_i)$. Note that atoms in $At(F) \setminus At(a_i \to F_i)$ do not occur in $im(a_i \to F_i)$ because the latter adds new fresh atoms to $At(F_i)$. Therefore, by Theorem 1, we can extend $(a_i \to F_i) \cong_{At(F_i)} im(a_i \to F_i)$ to a larger vocabulary and obtain $(a_i \to F_i) \cong_{At(F)} im(a_i \to F_i)$. As a result, we get $F \cong_{At(F)} im(F)$. With respect to size, we have:

$$\begin{aligned}
|im(F)| &= |\varphi| + 3m + \textstyle\sum_{i=1}^m |im(a_i \to F_i)| \\
&\leq |\varphi| + 3m + \textstyle\sum_{i=1}^m |a_i \to F_i|^2 \\
&= |\varphi| + 3m + \textstyle\sum_{i=1}^m \big(2 + |F_i|\big)^2 \\
&\leq \underbrace{|\varphi|^2}_{\geq|\varphi|} + 3m + 4m + \textstyle\sum_{i=1}^m \underbrace{2m}_{\geq 4} |F_i| + \sum_{i=1}^m |F_i|^2 \\
&\leq |\varphi|^2 + \underbrace{m^2 - 2m}_{\geq 0} + 7m + \textstyle\sum_{i=1}^m 2m|F_i| + \sum_{i=1}^m |F_i|^2 \\
&\leq |\varphi|^2 + m^2 + 2m + \textstyle\sum_{i=1}^m 2m|F_i| + \sum_{i=1}^m |F_i|^2 + m + 2m \\
&\leq |\varphi|^2 + m^2 + 2m|\varphi| + \textstyle\sum_{i=1}^m 2m|F_i| + \sum_{i=1}^m |F_i|^2 \\
&\quad + \underbrace{2 \sum_{1 \leq j < i \leq m} |F_j||F_i|}_{\geq 2\ (m\ (m-1)/2) \geq m} + \underbrace{\sum_{i=1}^{m} 2|\varphi||F_i|}_{\geq 2m} \\
&= |\varphi|^2 + m^2 + \big(\textstyle\sum_{i=1}^m |F_i|\big)^2 \\
&\quad + 2m|\varphi| + \textstyle\sum_{i=1}^m 2m|F_i| + \sum_{i=1}^m 2|\varphi||F_i| \\
&= \big(|\varphi| + m + \textstyle\sum_{i=1}^m |F_i|\big)^2 \\
&= |F|^2
\end{aligned}$$

Note that $|\varphi| \geq 1$, $|F_i| \geq 1$ and $m \geq 2$.

- In case $F = \varphi \to (F_1 \wedge \ldots \wedge F_m)$, we get $F \cong (\varphi \to F_1) \wedge \ldots \wedge (\varphi \to F_m)$ by (4). Furthermore, from Lemma 3, we get that

$$(\varphi \to F_i) \cong_V (\varphi \to a) \wedge (a \to F_i)$$

for any set of atoms V such that $a \notin V$. Therefore, we can say that

$$F \cong_{At(F)} (\varphi \to a) \wedge (a \to F_1) \wedge \ldots \wedge (a \to F_m)$$

We also have $\delta(a \to F_i) < \delta(F)$, and, thus, the result follows by the induction hypothesis. As for the size, note that

$$\begin{aligned}
& |im(F)| \\
={}& |\varphi| + 2 + m + \textstyle\sum_{i=1}^m |im(a \to F_i)| \\
\leq{}& |\varphi| + 2 + m + \textstyle\sum_{i=1}^m |a \to F_i|^2 \\
={}& |\varphi| + 2 + \textstyle\sum_{i=1}^m \big(|F_i| + 2\big)^2 + m \\
={}& |\varphi| + 2 + \textstyle\sum_{i=1}^m \big(|F_i|^2 + 4|F_i| + 4\big) + m \\
={}& |\varphi| + 2 + \textstyle\sum_{i=1}^m |F_i|^2 + 4\sum_{i=1}^m |F_i| + 4m + m \\
\leq{}& \underbrace{|\varphi|^2}_{\geq|\varphi|} + \underbrace{m^2}_{\geq m \geq 2} + \textstyle\sum_{i=1}^m |F_i|^2 + \underbrace{2m}_{\geq 4} \sum_{i=1}^m |F_i| \\
& \quad + \underbrace{2|\varphi|m}_{\geq 2m} + \underbrace{2|\varphi| \sum_{i=1}^{m} |F_i|}_{\geq 2m} + \underbrace{2 \sum_{1 \leq j < i \leq m} |F_j||F_i|}_{\geq 2\ (m\ (m-1)/2) \geq m} \\
={}& |\varphi|^2 + m^2 + \big(\textstyle\sum_{i=1}^m |F_i|\big)^2 + 2|\varphi|m + 2m\sum_{i=1}^m |F_i| + 2|\varphi|\sum_{i=1}^m |F_i| \\
={}& \big(|\varphi| + m + \textstyle\sum_{i=1}^m |F_i|\big)^2 \\
={}& |F|^2
\end{aligned}$$

- If $F = (\varphi \to (\psi \to G))$, by (5), we get $F \cong \psi \wedge \varphi \to G$. Furthermore,

$$\delta(\psi \wedge \varphi \to G) = 1 + \delta(G) \quad < \quad \delta(F) = 2 + \delta(G)$$

The size does not increase because $|im(\varphi \to (\psi \to G))|$ is equal to:

$$|im(\varphi \wedge \psi \to G)| \leq |(\varphi \wedge \psi \to G)|^2 = |(\varphi \to (\psi \to G))|^2.$$

Finally, it is easy to see that every recursive step can be computed in polynomial time and that the number of recursive calls is bounded by the size of the fork. □

Once the main properties have been guaranteed for $im(\cdot)$, the proof for $pf(\cdot)$ follows almost immediately.

Proof of the Main Theorem. The cases where F is a propositional formula or $F = \varphi \to G$ directly follow from the previous Theorem 3. Otherwise, we get $|F| \geq 3$ (because F is not a propositional formula). Furthermore, $F \neq (\varphi \to G)$ implies $pf(F) = im(\top \to F) \cong_V (\top \to F) \cong F$ by (6). Finally:

$$|pf(F)| = |im(\top \to F)| \leq |\top \to F|^2 = (2 + |F|)^2 \leq 3|F|^2$$

since $|F| \geq 3$. □

As mentioned above, $pf(F)$ does not always produce a logic program: to see why, it suffices to observe that $pf(\varphi) = \varphi$ for any arbitrary propositional formula like, say, $pf((p \to q) \vee r) = (p \to q) \vee r$. There exist several methods in the literature for reducing propositional formulas to (disjunctive) logic programs under the stable model semantics. In particular, the already mentioned reduction in [14] is polynomial and strongly faithful.[6] Given that the complexity for brave and cautious reasoning for disjunctive programs are Σ_2^P and Π_2^P-complete, respectively [15], we immediately conclude:

Corollary 2. *Brave and cautions reasoning for (arbitrary) forks are Σ_2^P and Π_2^P-complete, respectively.* □

5. Conclusions

This research note extends an earlier published paper [1], where we studied projective strong equivalence (PSE) of logic programs and introduced a new logical connective called “fork.” Although forgetting auxiliary atoms is not always possible in ASP [5], we proved that this impossibility is removed when we admit programs with forks. This result justified the theoretical interest of this new connective, but its practical application was somehow limited by the fact that the translation to implement forks back as regular logic programs (adding new fresh auxiliary atoms) presented in [1] had exponential size in the worst case. In this note, we have provided a new translation that satisfies PSE and has, at most, a quadratic size. This allowed us to prove that brave and cautious reasoning with forks has the same complexity that of disjunctive logic programs. Besides, it paves the way for an efficient implementation of the fork connective using ASP solvers.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We wish to thank the anonymous reviewers for their useful suggestions that have helped to improve the paper. This work was partially supported by MICINN, Spain, grant PID2020-116201GB-I00, Xunta de Galicia, Spain, grant GPC ED431B 2019/03, Universidade da Coruña/CISUG, Spain, (funding for open access charge) and National Science Foundation, USA, grant NSF Nebraska EPSCoR 95-3101-0060-402.

References

[1] F. Aguado, P. Cabalar, J. Fandinno, D. Pearce, G. Pérez, C. Vidal, Forgetting auxiliary atoms in forks, Artif. Intell. 275 (2019) 575–601.
[2] C. Baral, Knowledge Representation, Reasoning and Declarative Problem Solving, Cambridge University Press, 2003.
[3] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R. Kowalski, K. Bowen (Eds.), Proceedings of the 19th International Conference and Symposium of Logic Programming (ICLP'88), MIT Press, 1988, pp. 1070–1080.
[4] D. Pearce, A new logical characterisation of stable models and answer sets, in: J. Dix, L.M. Pereira, T.C. Przymusinski (Eds.), Selected Papers from the Non-Monotonic Extensions of Logic Programming (NMELP'96), in: Lecture Notes in Artificial Intelligence, vol. 1216, Springer-Verlag, 1996, pp. 57–70.
[5] R. Gonçalves, M. Knorr, J. Leite, You can't always forget what you want: on the limits of forgetting in answer set programming, in: G.A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum, F. van Harmelen (Eds.), Proceedings of 22nd European Conference on Artificial Intelligence (ECAI'16), in: Frontiers in Artificial Intelligence and Applications, vol. 285, IOS Press, 2016, pp. 957–965.
[6] V. Lifschitz, D. Pearce, A. Valverde, Strongly equivalent logic programs, ACM Trans. Comput. Log. 2 (4) (2001) 526–541.
[7] T. Eiter, H. Tompits, S. Woltran, On solution correspondences in answer-set programming, in: L.P. Kaelbling, A. Saffiotti (Eds.), Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05), Professional Book Center, 2005, pp. 97–102.

[6] To be precise, this reduction obtains disjunctive programs with negation in the head, but the latter can be, in its turn, replaced by auxiliary atoms in linear time.

[8] A. Heyting, Die formalen Regeln der intuitionistischen Logik, Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse, 1930, pp. 42–56.
[9] J. Łukasiewicz, Die logik und das grundlagenproblem, Institut International de Cooperation Intellectuelle, 1941.
[10] D. Pearce, V. Sarsakov, T. Schaub, H. Tompits, S. Woltran, A polynomial translation of logic programs with nested expressions into disjunctive logic programs: preliminary report, in: P.J. Stuckey (Ed.), Proc. of the 18th Intl. Conf. of Logic Programming, ICLP 2002, Copenhagen, Denmark, July 29 - August 1, 2002, Proceedings, in: Lecture Notes in Computer Science, vol. 2401, Springer, 2002, pp. 405–420.
[11] T. Eiter, M. Fink, S. Woltran, Semantical characterizations and complexity of equivalences in answer set programming, ACM Trans. Comput. Log. 8 (3) (2007) 17.
[12] K.S. Wong, Forgetting in logic programs, Ph.D. thesis, The University of New South Wales, 2009.
[13] R. Gonçalves, M. Knorr, J. Leite, The ultimate guide to forgetting in answer set programming, in: Proc. of the 15th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR'16), AAAI Press, 2016, pp. 135–144.
[14] P. Cabalar, D. Pearce, A. Valverde, Reducing propositional theories in equilibrium logic to logic programs, in: C. Bento, A. Cardoso, G. Dias (Eds.), Proceedings of the 12th Portuguese Conference on Progress in Artificial Intelligence (EPIA'05), in: Lecture Notes in Computer Science, vol. 3808, Springer, 2005, pp. 4–17.
[15] T. Eiter, G. Gottlob, H. Mannila, Expressive power and complexity of disjunctive datalog under the stable model semantics, in: K. von Luck, H. Marburger (Eds.), Management and Processing of Complex Data Structures, Third Workshop on Information Systems and Artificial Intelligence, Hamburg, Germany, February 28 - March 2, 1994, Proceedings, in: Lecture Notes in Computer Science, vol. 777, Springer, 1994, pp. 83–103.