TPLP: Page 1–8. © The Author(s), 2025. Published by Cambridge University Press 2025
doi:10.1017/xxxxx

Comparing Non-minimal Semantics for Disjunction in Answer Set Programming

FELICIDAD AGUADO[1]
(*e-mail:* felicidad.aguado@udc.es)

PEDRO CABALAR[1]
(*e-mail:* cabalar@udc.es)

BRAIS MUÑIZ[1]
(*e-mail:* brais.mcastro@udc.es)

GILBERTO PÉREZ[1]
(*e-mail:* gperez@udc.es)

CONCEPCIÓN VIDAL[1]
(*e-mail:* concepcion.vidalm@udc.es)

[1] - *University of A Coruña, Spain*

submitted xx xx xxxx; revised xx xx xxxx; accepted xx xx xxxx

Abstract

In this paper, we compare four different semantics for disjunction in Answer Set Programming that, unlike stable models, do not adhere to the principle of model minimality. Two of these approaches, Cabalar and Muñiz' *Justified Models* and Doherty and Szalas' *Strongly Supported Models*, directly provide an alternative non-minimal semantics for disjunction. The other two, Aguado et al's *Forks* and Shen and Eiter's *Determining Inference* (DI) semantics, actually introduce a new disjunction connective, but are compared here as if they constituted new semantics for the standard disjunction operator. We are able to prove that three of these approaches (Forks, Justified Models and a reasonable relaxation of the DI semantics) actually coincide, constituting a common single approach under different definitions. Moreover, this common semantics always provides a superset of the stable models of a program (in fact, modulo any context) and is strictly stronger than the fourth approach (Strongly Supported Models), that actually treats disjunctions as in classical logic.

KEYWORDS: Answer Set Programming, Disjunctive Logic Programming, Equilibrium Logic, Forks

1 Introduction

Answer Set Programming (ASP) Marek and Truszczyński (1999); Niemelä (1999) constitutes nowadays a successful paradigm for practical Knowledge Representation and problem solving. Great part of this success is due to the rich expressiveness of the ASP language and its declarative semantics, based on the concept of *stable models* in Logic Programming (LP) proposed by Gelfond and Lifschitz 1988. Stable models were originally defined for normal logic programs, but later generalised to accommodate multiple

syntactic extensions. One of the oldest of such extensions is the use of disjunction in the rule heads Gelfond and Lifschitz (1991). Informally speaking, we may say that the extension of stable models to disjunctive logic programs is based on an *extrapolation of model minimality*. To explain this claim, let us first recall their original definition for the non-disjunctive case. To define a stable model I of a program P, we first obtain the so-called *program reduct* P^I, a program that corresponds to replacing each negative literal in P by its truth value according to I. Program P^I amounts to a set of definite Horn clauses and the semantics for these programs was well-established since the origins of LP. A (consistent) definite program always has a least model van Emden and Kowalski (1976) that further coincides with the the least fixpoint of the *immediate consequences* operator T_P, a derivation function that informally corresponds to an exhaustive application of Modus Ponens on the program rules. A model I of P is *stable* if it coincides with the least model of P^I or, equivalently, the least fixpoint of T_{P^I}. Now, once we introduce disjunction in the rule heads of P, the reduct P^I need not be a definite program any more. As a result, there is no guarantee of a least model (we may have several minimal ones) whereas operator T_{P^I} is not defined, since the application of Modus Ponens may not result in the derivation of atoms[1]. Therefore, two choices are available: (i) requiring I to be *one of the minimal models* of P^I; or (ii) modifying the way in which atoms in P^I can be derived, with some alternative to T_{P^I}. As a simple example, consider the disjunctive program $P_{(1)}$ consisting of rules:

$$a \vee b \qquad\qquad a \vee c \tag{1}$$

$P_{(1)}$ has five classical models $\{a\}$, $\{b, c\}$, $\{a, b\}$, $\{a, c\}$ and $\{a, b, c\}$ but only the first two are minimal. On the other hand, even though it is a positive program, the application of $T_{P_{(1)}}$ is undefined and the way to extend it for disjunctive heads is unclear. Stable models for disjunctive programs Gelfond and Lifschitz (1991) adopt criterion (i) based on "minimality" – it is surely the most natural option, but also introduces some drawbacks. First, we no longer have an associated derivation method like the immediate consequences operator used before. Second, the complexity of existence of stable model jumps one level in the polynomial hierarchy, from NP-complete Marek and Truszczynski (1991) for normal programs to Σ_2^P-complete for disjunctive programs Eiter and Gottlob (1995).

Alternative (ii) has also been explored in the literature in various ways, leading to different disjunctive LP semantics that do not adhere to minimality. Without trying to be exhaustive, we study here four alternatives that, despite coming from different perspectives, show stunning resemblances. These four approaches are (by chronological order) the *strongly supported models* by Doherty and Szałas (2015), the so-called *fork* operators by Aguado et al. (2019), the *determining inference (DI) semantics* by Shen and Eiter (2019) the same year, and the *justified models* by Cabalar and Muñiz (2024). In the paper, we prove that the last three cases actually coincide (with a slight relaxation of the DI-semantics), whereas strongly supported models constitute a strictly weaker semantics.

The rest of the paper is organised as follows. The background section contains a description of the approach based on forks which we will take as a reference for most of the

[1] It is still possible to derive sets of disjunctions of atoms Lobo et al. (1992), or sets of minimal interpretations Fernández and Minker (1995) but these options were less explored in the ASP literature.

correspondence proofs. It also contains a pair of new results (Section 2.3) about replacing disjunctions by forks. Section 3 describes justified models and proceeds then to prove that the stable models of a fork-based disjunctive program coincide with the justified models of a disjunctive logic program. In Section 4, we recall the DI-semantics and then prove that, under certain reasonable relaxations of this approach, it also coincides with the semantics of forks. The next section covers the case of strongly supported models, proving in this case that it constitutes a strictly weaker semantics with respect to the other three approaches, equivalent among them. Finally, Section 6 concludes the paper. The appendices contain the proofs and some previous additional definitions.

2 Background: overview of Forks

In this section, we revisit the basic definitions for the fork operators and their denotational semantics. This semantics is based in its turn on *Equilibrium Logic* Pearce (1996) and its monotonic basis, the logic of *Here-and-There* (HT) Heyting (1930), which is introduced in the first place. Then, we recall the definition of forks and some previous results that will be used later on for the proofs of correspondence with the other approaches. Finally, we conclude the section providing a new theorem (Th. 2) to be used later, that proves that the replacement of a disjunction by a fork in any arbitrary disjunctive logic program always produces a superset of stable models.

2.1 Here-and-There and Equilibrium Logic

Let $\mathcal{AT}$ be a finite set of atoms called the *alphabet* or *vocabulary*. A *(propositional) formula* φ 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 $p \in \mathcal{AT}$. We use Greek letters φ, ψ, γ and their variants to stand for formulas. We also define the derived operators $(\psi \leftarrow \varphi) \stackrel{\text{def}}{=} (\varphi \rightarrow \psi)$, $\neg\varphi \stackrel{\text{def}}{=} (\varphi \rightarrow \bot)$ and $\top \stackrel{\text{def}}{=} \neg\bot$. Given a formula φ, by $\mathcal{AT}(\varphi) \subseteq \mathcal{AT}$ we denote the set of atoms occurring in φ. A *theory* Γ is a finite[2] set of formulas that can be also understood as their conjunction. When a theory consists of a single formula $\Gamma = \{\varphi\}$ we will frequently omit the braces. An *extended disjunctive rule* r is an implication of the form:

$$p_1 \vee \cdots \vee p_m \ \leftarrow \ p_{m+1} \wedge \cdots \wedge p_n \wedge \neg p_{n+1} \wedge \cdots \wedge \neg p_h \wedge \neg\neg p_{h+1} \wedge \cdots \wedge \neg\neg p_k \quad (2)$$

where all p_i above are atoms in $\mathcal{AT}$ and $0 \leq m \leq n \leq h \leq k$. The disjunction in the consequent is called the *head* of r and denoted as $Head(r)$, whereas the conjunction in the antecedent receives the name of *body* of r and is denoted by $Body(r)$. We define the sets of atoms $h(r) \stackrel{\text{def}}{=} \{p_1, \ldots, p_m\}$, $b^+(r) \stackrel{\text{def}}{=} \{p_{m+1}, \ldots, p_n\}$, $b^-(r) \stackrel{\text{def}}{=} \{p_{n+1}, \ldots, p_h\}$, $b^{--}(r) \stackrel{\text{def}}{=} \{p_{h+1}, \ldots, p_k\}$ and $b(r) \stackrel{\text{def}}{=} b^+(r) \cup b^-(r) \cup b^{--}(r)$. We say that r is an *extended normal rule* if $|h(r)| \leq 1$. We drop the adjective "extended" when the rule does not have double negation. That is, when $k = h$ we simply talk about a *disjunctive rule* and further

[2] In this paper, we exclusively focus on finite theories since some of the semantics are not defined for the infinite case. We leave for future work studying which semantic relations are preserved in that case.

call it *normal rule*, if it satisfies $|h(r)| \leq 1$. An empty head $h(r) = \emptyset$ represents falsum $\bot$ and, when this happens, the rule is called a *constraint*. An empty body $b(r) = \emptyset$ is assumed to represent $\top$ and, when this happens, we usually omit $\leftarrow \top$ simply writing the rule head. A rule with $b(r) = \emptyset$ and $|h(r)| = 1$ is called a *fact*. A program P is a set of rules and, when the program is finite, we will also understand it as their conjunction. We say that program P belongs to a syntactic category if all its rules belong to that category.

A *classical interpretation* T is a set of atoms $T \subseteq \mathcal{AT}$. By $T \models \varphi$ we mean the usual classical satisfaction of a formula φ. Moreover, we write $M(\varphi)$ to stand for the set of classical models of φ. An HT-*interpretation* is a pair $\langle H, T \rangle$ (respectively called "here" and "there") of sets of atoms $H \subseteq T \subseteq \mathcal{AT}$; it is said to be *total* when $H = T$. Intuitively, an atom p is considered *false*, when $p \notin T$, or *true* when $p \in T$, but the latter has two cases: it may be *certainly true* when $p \in H$ or just *assumed true* when $p \in T \setminus H$. An interpretation $\langle H, T \rangle$ *satisfies* a formula φ, written $\langle H, T \rangle \models \varphi$, when the following recursive rules hold:

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

An HT-interpretation $\langle H, T \rangle$ is a *model* of a theory Γ if $\langle H, T \rangle \models \varphi$ for all $\varphi \in \Gamma$. Two formulas (or theories) φ and ψ are HT-equivalent, written $\varphi \equiv \psi$, if they have the same HT-models.

A total interpretation $\langle T, T \rangle$ is an *equilibrium model* of a formula φ iff $\langle T, T \rangle \models \varphi$ and there is no $H \subset T$ such that $\langle H, T \rangle \models \varphi$. If so, we say that T is a *stable model* of φ and we write $SM(\varphi)$ to stand for the set of stable models of φ.

2.2 Forks

A *fork* F is defined by the following grammar:

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

where φ is a propositional formula over $\mathcal{AT}$ and $p \in \mathcal{AT}$ is an atom. It can be proved, by structural induction, that any propositional formula φ is a fork. Note that a fork is not allowed as an argument of a disjunction nor as the antecedent of an implication. The intuition of this new connective '|' is that the stable models of a fork such as $(\varphi_1 \mid \cdots \mid \varphi_n)$ – in fact, all forks are reducible to this form – will be the *union* of stable models of each φ_i. The formal semantics of forks is based on the idea of *denotations* (sets of models) we define next in several steps.

Given a set of atoms $T \subseteq \mathcal{AT}$, a T-*support* $\mathcal{H} \subseteq 2^T$ is a set of subsets of T so that, if $\mathcal{H} \neq \emptyset$, then $T \in \mathcal{H}$. Given a propositional formula φ, the set of sets of atoms $\{H \subseteq T \mid \langle H, T \rangle \models \varphi\}$ forms a T-support we denote as $[\![\varphi]\!]^T$. For readability sake, we directly write a T-support as a sequence of sets between square braces: for instance, some possible supports for $T = \{a, b\}$ are $[\{a, b\}\ \{a\}]$, $[\{a, b\}\ \{b\}\ \emptyset]$ or the empty support $[\ ]$. Given two T-supports, $\mathcal{H}$ and $\mathcal{H}'$, we define the order relation $\mathcal{H} \preceq \mathcal{H}'$ iff either $\mathcal{H} = [\ ]$

or $[\,] \neq \mathcal{H}' \subseteq \mathcal{H}$, read as $\mathcal{H}$ is "less supported" than $\mathcal{H}'$. Intuitively, this means that $\mathcal{H}'$ is closer to make T a stable model than $\mathcal{H}$. Given a T-support $\mathcal{H}$, we define its complementary support $\overline{\mathcal{H}}$ as:

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

The *ideal* of $\mathcal{H}$ is defined as $\downarrow\mathcal{H} = \{\mathcal{H}' \mid \mathcal{H}' \preceq \mathcal{H}\} \setminus \{\,[\,]\,\}$. Note that, the empty support $[\,]$ is not included in the ideal, so $\downarrow[\,] = \emptyset$. If Δ is any set of supports, we use its $\preceq$-closure:

$$^{\downarrow}\Delta \stackrel{\text{def}}{=} \bigcup_{\mathcal{H}\in\Delta} \downarrow\mathcal{H} = \bigcup_{\mathcal{H}\in\Delta} \{\,\mathcal{H}' \preceq \mathcal{H} \mid \mathcal{H}' \neq [\,]\,\}.$$

We define a T-*view* Δ as any $\preceq$-closed set of T-supports, i.e., $^{\downarrow}\Delta = \Delta$. Given a T-view Δ, we write $\mathcal{H}\,\hat{\in}\,\Delta$ iff $\mathcal{H} \in \Delta$ or both $\mathcal{H} = [\,]$ and $\Delta = \emptyset$.

Definition 1 (T**-denotation)**
Let $\mathcal{AT}$ be a propositional signature and $T \subseteq \mathcal{AT}$ a set of atoms. The T-*denotation* of a fork or a propositional formula F, written $\langle\!\langle\, F \,\rangle\!\rangle^T$, is a T-view recursively defined as follows:

$$\begin{array}{rcl}
\langle\!\langle\, \bot \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & \emptyset \\
\langle\!\langle\, p \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & \downarrow[\![\, p \,]\!]^T \quad \text{for any atom } p \\
\langle\!\langle\, F \wedge G \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & ^{\downarrow}\{\, \mathcal{H} \cap \mathcal{H}' \mid \mathcal{H} \in \langle\!\langle\, F \,\rangle\!\rangle^T \text{ and } \mathcal{H}' \in \langle\!\langle\, G \,\rangle\!\rangle^T \,\} \\
\langle\!\langle\, \varphi \vee \psi \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & ^{\downarrow}\{\, \mathcal{H} \cup \mathcal{H}' \mid \mathcal{H}\,\hat{\in}\,\langle\!\langle\, \varphi \,\rangle\!\rangle^T \text{ and } \mathcal{H}'\,\hat{\in}\,\langle\!\langle\, \psi \,\rangle\!\rangle^T \,\} \\
\langle\!\langle\, \varphi \to F \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & \begin{cases} \{2^T\} & \text{if } [\![\, \varphi \,]\!]^T = [\,] \\ ^{\downarrow}\{\, \overline{[\![\, \varphi \,]\!]^T} \cup \mathcal{H} \mid \mathcal{H} \in \langle\!\langle\, F \,\rangle\!\rangle^T \,\} & \text{otherwise} \end{cases} \\
\langle\!\langle\, F \mid G \,\rangle\!\rangle^T & \stackrel{\text{def}}{=} & \langle\!\langle\, F \,\rangle\!\rangle^T \cup \langle\!\langle\, G \,\rangle\!\rangle^T
\end{array}$$

where F, G denote forks or propositional formulas.

We say that T is a *stable model* of a fork F when $\langle\!\langle\, F \,\rangle\!\rangle^T = \downarrow[\,T\,]$ or, equivalently, when $[\,T\,] \in \langle\!\langle\, F \,\rangle\!\rangle^T$. The set $SM(F)$ collects all the stable models of F.

Definition 2 (*Strong Entailment/Equivalence of forks*)
We say that fork F *strongly entails* fork G, in symbols $F \mathrel{|\!\sim} G$, if $SM(F \wedge L) \subseteq SM(G \wedge L)$, for any fork L. We further say that F and G *are strongly equivalent*, in symbols $F \cong G$ if both $F \mathrel{|\!\sim} G$ and $G \mathrel{|\!\sim} F$, that is, $SM(F \wedge L) = SM(G \wedge L)$, for any fork L.

Interestingly, Aguado et al. (2019) (Prop. 11) proved that $F \mathrel{|\!\sim} G$ is equivalent to $\langle\!\langle\, F \,\rangle\!\rangle^T \subseteq \langle\!\langle\, G \,\rangle\!\rangle^T$, for every set of atoms $T \subseteq \mathcal{AT}$ and, thus, $F \cong G$ amounts to $\langle\!\langle\, F \,\rangle\!\rangle^T = \langle\!\langle\, G \,\rangle\!\rangle^T$, for every T. Other properties proved by Aguado et al. (2019) we will use below are:

$$(F \mid G) \mid L \;\cong\; F \mid (G \mid L) \tag{3}$$

$$(F \mid G) \wedge L \;\cong\; (F \wedge L) \mid (G \wedge L) \tag{4}$$

$$SM(F \mid G) \;=\; SM(F) \cup SM(G) \tag{5}$$

Example 1

Consider the fork:

$$(a \mid b) \wedge (a \mid c) \tag{6}$$

We can apply distributivity (4) and associativity (3) to conclude that (6) is actually strongly equivalent to:

$$a \wedge a \mid a \wedge c \mid b \wedge a \mid b \wedge c$$

which is a fork built with 4 propositional formulas. By (5), the stable models of this fork are the union of stable models of these 4 formulas, namely, $\{a\}$, $\{a, c\}$, $\{a, b\}$ and $\{b, c\}$. □

We conclude this section introducing the polynomial reduction of any fork F into a propositional formula $pf(F)$ by Aguado et al. (2022) that may help for a better understanding of the behaviour of forks, and is used in the proof of Theorem 4 later on. For simplicity, we constrained here $pf(F)$ to the case in which F has the form $P^|$ for some extended disjunctive program P, using less definitions and getting $pf(F)$ in the form of a disjunctive logic program.

Definition 3
Let P be some extended disjunctive logic program. For each $r \in P$ we define $pf(r^|)$ as: $pf(r^|) \stackrel{\text{def}}{=} r$ if r is an extended normal rule, i.e. $|h(r)| \leq 1$; otherwise, given $h(r) = \{p_1, \ldots, p_m\}$:

$$pf(r^|) \stackrel{\text{def}}{=} (x_1 \vee \cdots \vee x_m \leftarrow \mathit{Body}(r)) \ \wedge \ \bigwedge_{i=1}^{m} (p_i \leftarrow x_i)$$

for a set of fresh propositional atoms $x_1, \ldots, x_m$. □

We also define $pf(P^|) \stackrel{\text{def}}{=} \bigwedge_{r \in P} pf(r^|)$. E.g., $pf(P^|_{(1)})$ is the conjunction of:

$$x_1 \vee x_2 \qquad a \leftarrow x_1 \qquad b \leftarrow x_2 \qquad y_1 \vee y_2 \qquad a \leftarrow y_1 \qquad c \leftarrow y_2$$

Theorem 1 (*From Main Theorem Aguado et al. (2022)*)
Let P be an extended logic program. $P^|$ and $pf(P^|)$ are strongly equivalent, modulo alphabet $\mathcal{AT}(P)$. □

2.3 Replacing disjunctions by forks

As expected, the definition of stable models for forks is a proper extension of stable models for propositional theories (or if preferred, equilibrium models Pearce (1996)) and so, in its turn, it also applies to the more restricted syntax of logic programs with disjunction Gelfond and Lifschitz (1991). This means that disjunction '$\vee$' in logic programs respects the principle of minimality. For instance, under this definition we still have the same two stable models for program $P_{(1)}$, namely, $SM(P_{(1)}) = \{\{a\}, \{b, c\}\}$. However, minimality is lost if we replace '$\vee$' by '$|$', as illustrated next.

For any disjunctive rule r, let us denote by $r^|$ the fork obtained by substituting in $h(r)$ the operator '$\vee$' by '$|$' and let $P^| \stackrel{\text{def}}{=} \bigwedge_{r \in P} r^|$ for any program P as expected. For instance, the fork $P^|_{(1)}$ would correspond to (6) in Example 1 whose stable models were $\{a\}$, $\{b, c\}$, $\{a, b\}$ and $\{a, c\}$ – the last two are not minimal whereas the first two coincide

with $SM(P_{(1)})$. The main result in this section proves that the replacement of regular disjunctions by forks in any rule r always produces a superset of stable models, even if that rule is included in a larger arbitrary context. Namely, we have the strong entailment relation $r \mid\!\sim r^{|}$.

Theorem 2
Let φ and $\alpha_1, \ldots, \alpha_n$ be propositional formulas with $n \geq 1$. Then:

$$\varphi \to (\alpha_1 \vee \cdots \vee \alpha_n) \;\mid\!\sim\; \varphi \to (\alpha_1 \mid \cdots \mid \alpha_n) \qquad \square$$

Since strong entailment allows us to proceed rule by rule, we conclude:

Corollary 1
Let P be any extended disjunctive logic program, then $P \mid\!\sim P^{|}$. $\square$

As a result, a disjunctive program that has no stable models may restore coherence (existence of stable model) if we replace disjunctions by forks. Take the following example (adapted from Ex. 1 by Shen and Eiter (2019),).

Example 2
Consider the program $P_{(7)}$ consisting of the three rules:

$$a \vee b \qquad\qquad a \qquad\qquad b \leftarrow \neg b \tag{7}$$

Disjunction $a \vee b$ is redundant and can be removed, because it is an HT-consequence of a. But once $a \vee b$ disappears, it is clear that $b \leftarrow \neg b$ prevents obtaining any stable model. Yet, if we change the disjunction in $a \vee b$ by a fork, we can restore coherence. The fork $P^{|}_{(7)}$ corresponds to:

$$\begin{array}{lll} & (a \mid b) \wedge a \wedge (\neg b \to b) & \\ \cong & (a \mid b) \wedge a \wedge \neg\neg b & \text{HT-equivalence} \\ \cong & (a \wedge a \wedge \neg\neg b) \mid (b \wedge a \wedge \neg\neg b) & \text{by distributivity (4)} \\ \cong & (a \wedge \neg\neg b) \mid (a \wedge b) & \text{HT-equivalence} \end{array}$$

and then $SM(P_{(7)}) = SM(a \wedge \neg\neg b) \cup SM(a \wedge b) = \emptyset \cup \{\{a, b\}\} = \{\{a, b\}\}$, so $P^{|}_{(7)}$ has a unique stable model $\{a, b\}$.

3 Justified Models

We proceed now to compare the forks semantics with *justified models* Cabalar and Muñiz (2024). This approach was originally introduced to provide a definition of *explanations* for the stable models of a logic program. Such explanations have the form of graphs built with rule labels and reflect the derivation of atoms in the model. A classical model of a logic program is said to be *justified* if it admits at least one of these explanation graphs. In the case of normal logic programs, justified and stable models coincide, but Cabalar and Muñiz (2024) observed that, when the program is disjunctive, it may have more justified models than stable models. In other words, although every stable model of a disjunctive program admits an explanation, we may have classical models of the program that admit an explanation but are not stable, breaking the principle of minimality in many cases. In this way, justified models provide a weaker semantics for disjunctive programs that, as

we will see, actually coincides with the behaviour of fork-based disjunction. Let us start recalling some basic definitions, examples and results by Cabalar and Muñiz (2024).

Definition 4 (*Labelled logic program*)
A *labelled rule* r is an expression of the form $\ell : (2)$ where (2) is any extended disjunctive rule and ℓ is the rule *label*, we will also denote as $Lb(r) = \ell$. A *labelled logic program* P is a set of labelled rules that has no repeated labels, that is, for any pair of different rules $r, r' \in P$, $Lb(r) \neq Lb(r')$.

If r is a labelled rule, we keep the definitions of the formulas $Body(r)$ and $Head(r)$ and sets of atoms $h(r)$, $b(r)$, $b^+(r)$, $b^-(r)$ and $b^{--}(r)$ as before, that is, ignoring the additional label. Similarly, if P is a labelled logic program, $P^|$ denotes the fork that results from removing the labels and, as before, replacing disjunctions $\vee$ by $|$. A set of atoms I is a classical model of a labelled rule r iff $I \models Body(r) \rightarrow Head(r)$ in classical logic. Given a labelled logic program P, by $Lb(P)$ we denote the set of labels of the program $Lb(P) \stackrel{\text{def}}{=} \{Lb(r) \mid r \in P\}$. Note that no label is repeated, but P can contain two rules r, r' with the same body and head and different labels $Lb(r) \neq Lb(r')$. For instance, we could have two repeated facts with different labels $\ell_1 : p$ and $\ell_2 : p$ possibly representing two different and simultaneously applicable sources of information.

Definition 5 (*Support Graph/Explanation*)
Let P be a labelled program and I a classical model of P. A *support graph* G of I under P is a labelled directed graph $G = \langle I, E, \lambda \rangle$ where the vertices are the atoms in I, the (directed) edges $E \subseteq I \times I$ connect pairs of atoms, and $\lambda : I \rightarrow Lb(P)$ is an injective function that assigns a label for every atom $p \in I$ so that: if $r \in P$ is the rule with $Lb(r) = \lambda(p)$ then $p \in h(r)$, $I \models Body(r)$ and $b^+(r) = \{q \mid (q, p) \in E\}$. A support graph G is said to be an *explanation* if it additionally satisfies that G is acyclic. □

The fact that λ is injective guarantees that there are no repeated labels in the graph. Additionally, the definition tells us that if an atom p is labelled with $\lambda(p) = \ell$ then ℓ must be the label of some rule r where (1) p occurs in the head, (2) the body of the rule is satisfied by I and (3) the incoming edges for p are formed from the atoms in the positive body of r. Since an explanation $G = \langle I, E, \lambda \rangle$ for a model I is uniquely determined by its atom labelling λ, we can abbreviate G as a set of pairs $p \mapsto \lambda(p)$ for $p \in I$.

Definition 6 (*Supported/Justified model*)
Let I be classical model of a labelled program P, $I \models P$. Then, I is said to be a (*graph-based*) *supported model* of P if there exists some support graph of I under P, and is further said to be a *justified model* of P if there exists some explanation (i.e. acyclic support graph) of I under P. Sets $SPM(P)$ and $JM(P)$ respectively stand for the (graph-based) supported and justified models of P. □

We can also define $SPM(P)$ and $JM(P)$ for any non-labelled program P by assuming we previously label each rule in P with a unique arbitrary identifier. Note that different labels produce different explanations, but the definition of justified/supported model is not affected by that.

Theorem 3 (*Th. 1 and Th. 2 from Cabalar and Muñiz (2024)*)

If P is a labelled disjunctive program, then: $SM(P) \subseteq JM(P)$. Moreover, if P contains no disjunction, then $SM(P) = JM(P)$. □

However, if we allow disjunction, we may have justified models that *are not* stable models, as illustrated below.

Example 3
Let $P_{(8)}$ be the following labelled version of $P_{(1)}$:

$$\ell_1 : a \vee b \qquad\qquad \ell_2 : a \vee c \tag{8}$$

The classical models of $P_{(8)}$ are $\{a\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}$. The last one, $\{a, b, c\}$, is not justified, since we would need three different labels and we only have two rules. Each model $\{a, c\}$, $\{a, b\}$, $\{b, c\}$ has a unique explanation corresponding to the atom labellings $\{a \mapsto \ell_1, c \mapsto \ell_2\}$, $\{b \mapsto \ell_1, a \mapsto \ell_2\}$ and $\{b \mapsto \ell_1, c \mapsto \ell_2\}$, respectively. On the other hand, model $\{a\}$ has two possible explanations, corresponding to $\{a \mapsto \ell_1\}$ and $\{a \mapsto \ell_2\}$. To sum up, we get four justified models, $\{a, c\}$, $\{a, b\}$, $\{b, c\}$ and $\{a\}$ but only two of them are stable, $\{a\}$ and $\{b, c\}$. □

In other words, the justified models of $P_{(8)}$ coincide with the stable models of its fork version $P^{|}_{(8)} = P^{|}_{(1)} = (6)$ seen before. This is in fact, a general property that constitutes the main result of this section.

Theorem 4
$JM(P) = SM(P^{|})$ for any labelled disjunctive logic program P. □

Supported models $SPM(P)$ correspond to the case in which we also accept cyclic explanation graphs. Obviously, $JM(P) \subseteq SPM(P)$, because all acyclic explanations are still acceptable for $SPM(P)$. Cabalar and Muñiz (2024) also proved that $SPM(P)$ generalise the standard notion of supported models – i.e., models of Clark's completion Clark (1978) – to the disjunctive case. For instance, the program $P_{(9)}$ consisting of the rule:

$$\ell_1 : p \leftarrow p \tag{9}$$

has two supported models, $I = \emptyset$ (which is also stable and justified) and $I = \{p\}$ with a cyclic support graph where node p is connected to itself. As a remark, notice that the definition of our "graph-based" *supported models* Cabalar and Muñiz (2023) does not correspond to the (also called) supported models obtained from the *program completion* defined by Alviano and Dodaro (2016) for disjunctive programs. The latter, we denote $AD(P)$, impose a stronger condition: a rule r supports an atom $p \in Hd(r)$ with respect to interpretation I not only if $I \models Body(r)$ but also $I \not\models q$ for all $q \in Hd(r) \setminus p$. To illustrate the difference, take program $P_{(8)}$: as it has no cyclic dependencies, graph-based supported and justified models coincide, that is, $SPM(P_{(8)}) = JM(P_{(8)}) = \{\{a\}, \{a, b\}, \{a, c\}, \{b, c\}\}$ we saw before. However, $AD(P_{(8)}) = \{\{a\}, \{b, c\}\}$ that, in this case, coincide with the stable models of the program. Model $\{a, b\}$ is supported (under Def, 6) because a is justified by rule ℓ_1 and b by rule ℓ_2. However, under Alviano and Dodaro's definition, rule ℓ_1 is not a valid support for a since we would further need b (the other atom in the disjunction ℓ_1) to be false. The situation for model $\{a, c\}$ is analogous. It is not hard to see that $SM(P) \subsetneq AD(P) \subsetneq SPM(P)$ (the first inclusion proved by Alviano and Dodaro (2016)), so clearly, $AD(P)$ is more interesting for computation purposes when our goal is approximating $SM(P)$. However, $SPM(P)$ provides a more

liberal generalisation of the definition of supported model from normal logic programs: as in that case, I is a supported model of P if, for every atom $p \in I$, there exists some rule r with p "in the head" and $I \models Body(r)$. This definition has also a closer relation to $JM(P)$ and explanation generation or to the DI semantics (as we see in Theorem 8 in the next section).

4 Determining Inference

The third approach we consider, *Determining Inference* (DI) Shen and Eiter (2019), also introduces a new disjunction operator in rule heads, with the same syntax as forks '|'. Besides, first-order formulas are allowed to play the role of atoms, and so, the syntax accepts regular disjunction '$\vee$' too. However, in this paper (for the sake of comparison) we describe the DI-semantics directly on the syntax of extended disjunctive rules of the form (2) seen before[3], using '$\vee$' to play the role of the DI disjunctive operator.

The DI-semantics understands disjunction as a non-deterministic choice and is based on the definition of a *head selection function*. This function will tell us, beforehand, which head atom will be chosen if we have to apply a rule for derivation. We introduce next a slight generalisation of that definition.

Definition 7 (*Open/Closed Head Selection Function*)
Let P be an extended disjunctive logic program and $I \subseteq \mathcal{AT}$ an interpretation. A *head selection function sel* for I and some $r \in P$ is a formula:

$$sel(Head(r), I) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } h(r) \cap I = \emptyset \\ p_i & \text{otherwise, for some } p_i \in h(r) \cap I \end{cases}$$

We say that *sel* is *closed* if $sel(Head(r), I) = sel(Head(r'), I)$ for any pair of rules r, r' with the same head atoms $h(r) = h(r')$. If this restriction does not apply, we just say that *sel* is *open*. □

The original definition by Shen and Eiter (2019) (Def. 4) corresponds to what we call here *closed* selection function and forces the same choice when two rule heads are formed by the same set of atoms.

The *reduct* of a program P with respect to some interpretation I and selection function *sel* is defined as the logic program $P^I_{sel} \stackrel{\text{def}}{=} \{\ sel(Head(r), I) \leftarrow Body(r) \mid I \models Body(r)\ \}$. Note that P^I_{sel} is an extended normal logic program (possibly containing constraints) where we replaced each disjunction by the atom determined by the selection function *sel*.

Definition 8 (*Candidate stable model*)
A classical model I of a an extended disjunctive logic program P is said to be a *candidate stable model*[4] if there exists a selection function *sel* such that $I \in SM(P^I_{sel})$. We further say that I is *closed* if *sel* is closed. By $CSM(P)$, we denote the set of candidate stable models of P. □

[3] To be precise, Shen and Eiter (2019) treat double negation classically, whereas here, we take the liberty to keep doubly negated atoms in the body and interpret them as in Equilibrium Logic.
[4] Or *candidate answer set* in the original terminology Shen and Eiter (2019).

To understand the difference between closed and open selection functions, take the following program $P_{(10)}$:

$$p \qquad \bot \leftarrow c \qquad a \vee b \qquad b \vee a \leftarrow p \tag{10}$$

The set $CSM(P_{(10)})$ consists of $\{p, a\}$, $\{p, b\}$ and $\{p, a, b\}$, but only the first two models are closed, since they make the same choice in both disjunctions $a \vee b$ and $b \vee a$ that have the same atoms. Note that this condition is rather syntax-dependent: if we replace $b \vee a \leftarrow p$ by the rule $b \vee a \vee c \leftarrow p$, then open candidate stable models are not affected (c must be false due to constraint $\bot \leftarrow c$) but $\{p, a, b\}$ becomes now a closed candidate stable model, since the sets of atoms in $a \vee b$ and $b \vee a \vee c$ are different.

A *DI-stable model* I of a program P is a model that is minimal among the closed candidate stable models (Def. 7 by Shen and Eiter (2019)). Thus, DI-semantics actually imposes an additional minimality condition. However, if we focus on the previous step, $CSM(P)$, we can prove that they coincide with $SM(P^{|})$ and, by Theorem 4, with $JM(P)$ too.

Theorem 5
$CSM(P) = SM(P^{|})$ for any extended disjunctive logic program P. □

We conclude this section by proving that the decision problem $CSM(P) \neq \emptyset$ is NP-complete, recalling the following complexity result proved by Shen and Eiter (2019)

Proposition 6 (*From Table 1 by Shen and Eiter (2019)*)
Deciding the existence of a DI-stable model for a disjunctive program, under the well-justified semantics Shen et al. (2014), is an NP-complete problem.

Theorem 7
Given an extended disjunctive logic program P, deciding $CSM(P) \neq \emptyset$ is an NP-complete problem.

As one last result in this section, we provide an alternative characterisation of the supported models from Def. 6 using DI semantics. For normal logic programs, I is a supported model of P if, for every atom $p \in I$, there exists some rule r with p in the head and $I \models Body(r)$. Alternatively, supported models can also be captured as the fixpoints of the immediate consequences van Emden and Kowalski (1976) operator $T_P \stackrel{\text{def}}{=} \{p \mid (p \leftarrow B) \in P, I \models B\}$, namely, I is a supported model of P iff $I = T_P(I)$. We can extend this relation for disjunctive logic programs as follows.

Theorem 8
Let P be a labelled program and I a classical model of P. The following assertions are equivalent:

1. $I \in SPM(P)$
2. $T_{P^I_{sel}}(I) = I$ for some head selection function sel.

5 Strongly supported models

For our last comparison, we consider *strongly supported models* by Doherty and Szałas (2015):

Definition 9 (*Strongly Supported Models*[5])
A model T of an extended disjunctive logic program P is a *strongly supported model* of P if there exists a sequence of interpretations $H_0 \subseteq H_1 \subseteq \cdots \subseteq H_n = T$ such that

1. For $i = 0$: $H_0 \cap h(r) \neq \emptyset$ for all $r \in P$ with $b(r) = \emptyset$.
   For $i \geq 1$: $H_i \cap h(r) \neq \emptyset$ for all $r \in P$ with [6] $\langle H_{i-1}, T \rangle \models Body(r)$.
2. For each $i \geq 0$: H_i only contains atoms obtained by applying point 1, that is, if $p \in H_i$ then $p \in h(r)$ for some rule r mentioned in point 1.

We denote the set of strongly supported models of P as $SSM(P)$. □

Doherty and Szałas (2015) (Th. 1) proved that the stable models of P, $SM(P)$, coincide with the minimal elements of $SSM(P)$ and, furthermore, $SM(P) = SSM(P)$ when P has no disjunction. However, in general, the *SSM* semantics makes disjunction to behave classically. For instance, from Def. 9 above, we can easily observe that, if P is a set of disjunctions of atoms, then $SSM(P) = M(P)$. As a result, since (1) is a pair of disjunctions, $SSM(P_{(1)}) = M(P_{(1)})$ i.e., the five classical models $\{a\}$, $\{a, b\}$, $\{a, c\}$, $\{b, c\}$ and $\{a, b, c\}$ mentioned before. Note that *CSM* did not accept $\{a, b, c\}$, pointing our that it is a stronger semantics, as corroborated next:

Theorem 9
$CSM(P) \subseteq SSM(P)$ for any extended disjunctive logic program P. □

To conclude this section, we observe that, despite their name similarity, supported $SPM(P)$ and strongly supported models $SSM(P)$ are unrelated. To prove $SPM(P) \not\subseteq SSM(P)$, just take the program $P_{(9)}$ with no disjunctions, so that $SSM(P) = SM(P) = \{\emptyset\}$. However, $\{p\} \in SPM(P)$ as we discussed before. To prove $SSM(P) \not\subseteq SPM(P)$ we already saw that $\{a, b, c\} \in SSM(P_{(1)}) \setminus JM(P_{(1)})$. But, since $P_{(1)}$ has no implications, the support graphs contain no edges, so that acyclicity is irrelevant meaning $JM(P_{(1)}) = SPM(P_{(1)})$.

6 Conclusions

We have studied four different semantics for any disjunctive logic program P in Answer Set Programming (ASP) that, unlike the standard stable models $SM(P)$ do not adhere to the principle of model minimality. These four approaches are: *forks* Aguado et al. (2019) here denoted as $SM(P^{|})$; *justified models* Cabalar and Muñiz (2024) $JM(P)$; (a relaxed version of) *determining inference* Shen and Eiter (2019) we denoted $CSM(P)$; and *strongly supported models* $SSM(P)$ Doherty and Szałas (2015). The summary of our results is shown in Figure 1, where $M(P)$ represents the classical models of P and $SPM(P)$ an extension of supported models for the disjunctive case Cabalar and Muñiz (2024). Interestingly, the three semantics $SM(P^{|})$, $JM(P)$ and $CSM(P)$ coincide, although their definitions come from rather different approaches, showing that they may capture a significant way to understand disjunction in ASP, removing minimality and

[6] The original definition is not given in terms of HT-satisfaction, but it uses a definition involving pairs of sets of atoms that is completely equivalent, for the syntactic fragment of logic programs.

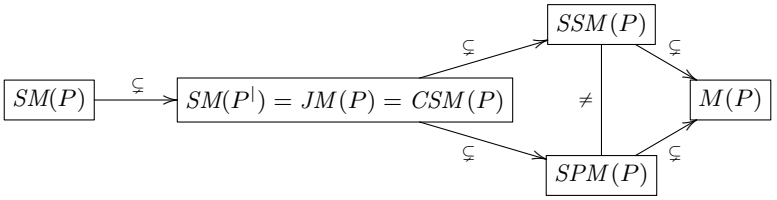

Fig. 1. Inclusion relations among several semantics for disjunctive logic programs.

keeping the computational complexity of existence of stable model as an NP-complete problem.

For future work, we plan to study other alternatives. For instance, one reviewer suggested replacing disjunctions by choice rules Simons et al. (2002) so that each disjunctive rule of the form $p_1 \vee \cdots \vee p_m \leftarrow \mathit{Body}$ becomes the choice rule $1\{p_1, \ldots, p_m\} \leftarrow \mathit{Body}$ and the rest of rules are left untouched. The behaviour of this replacement produces similar results to $SSM(P)$ and we plan to study a formal (dis)proof of this coincidence for future work.

Acknowledgements. We wish to thank the anonymous reviewers for their useful comments that have helped to improve the paper and, especially, for refuting an incorrect proof of a result included in a previous version of the document. This research was partially funded by the Spanish Ministry of Science, Innovation and Universities, MICIU/AEI/ 10.13039/501100011033, grant PID2023-148531NB-I00, Spain.

References

Aguado, F., Cabalar, P., Fandinno, J., Pearce, D., Pérez, G., and Vidal, C. 2019. Forgetting auxiliary atoms in forks. *Artificial Intelligence 275*, 575–601.

Aguado, F., Cabalar, P., Fandinno, J., Pearce, D., PÃ©rez, G., and Vidal, C. 2022. A polynomial reduction of forks into logic programs. *Artificial Intelligence 308*, 103712.

Alviano, M. and Dodaro, C. 2016. Completion of disjunctive logic programs. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, S. Kambhampati, Ed. IJCAI/AAAI Press, 886–892.

Cabalar, P. and Muñiz, B. 2023. Explanation graphs for stable models of labelled logic programs. In *Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP'23), CEUR Workshop Proceedings*. Vol. 3437.

Cabalar, P. and Muñiz, B. 2024. Model explanation via support graphs. *Theory and Practice of Logic Programming 24,* 6, 1109–1122.

Clark, K. L. 1978. Negation as failure. In *Logic and Databases*, H. Gallaire and J. Minker, Eds. Plenum, 293–322.

Doherty, P., Kvarnström, J., and Szalas, A. 2016. Iteratively-supported formulas and strongly supported models for kleene answer set programs - (extended abstract). In *Logics in Artificial Intelligence - 15th European Conference, JELIA 2016, Larnaca, Cyprus, November 9-11, 2016, Proceedings*, L. Michael and A. C. Kakas, Eds. Lecture Notes in Computer Science, vol. 10021. 536–542.

Doherty, P. and Szałas, A. 2015. *Stability, Supportedness, Minimality and Kleene Answer Set Programs*. Springer International Publishing, Cham, 125–140.

Eiter, T. and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence 15,* 3-4, 289–323.

Fernández, J. and Minker, J. 1995. Bottom-up computation of perfect models for disjunctive theories. *The Journal of Logic Programming 25,* 1, 33–51.

Gelfond, M. and Lifschitz, V. 1988. The Stable Model Semantics For Logic Programming. In *Proc. of the 5th International Conference on Logic Programming (ICLP'88)*. Seattle, Washington, 1070–1080.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9,* 3-4, 365–385.

Heyting, A. 1930. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften. Physikalisch-mathematische Klasse*.

Lobo, J., Minker, J., and Rajasekar, A. 1992. *Foundations of disjunctive logic programming.* Logic Programming. MIT Press.

Marek, V. and Truszczyński, M. 1999. *Stable models and an alternative logic programming paradigm.* Springer-Verlag, 169–181.

Marek, V. W. and Truszczynski, M. 1991. Autoepistemic logic. *Journal of the ACM 38,* 3, 588–619.

Niemelä, I. 1999. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3-4, 241–273.

Pearce, D. 1996. A New Logical Characterisation of Stable Models and Answer Sets. In *Proc. of Non-Monotonic Extensions of Logic Programming (NMELP'96)*. Bad Honnef, Germany, 57–70.

Shen, Y.-D. and Eiter, T. 2019. Determining inference semantics for disjunctive logic programs. *Artificial Intelligence 277*, 103–165.

Shen, Y.-D., Wang, K., Eiter, T., Fink, M., Redl, C., Krennwallner, T., and Deng, J. 2014. Flp answer set semantics without circular justifications for general logic programs. *Artificial Intelligence 213*, 1–41.

Simons, P., Niemelä, I., and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence 138,* 1, 181–234. Knowledge Representation and Logic Programming.

van Emden, M. H. and Kowalski, R. A. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM 23*, 733–742.

Appendix A Additional definitions

The proof of Theorem 4 requires some additional definitions by Aguado et al. (2019) and Cabalar and Muñiz (2023) that we include in this section. We start introducing some concepts for the denotational semantics that allow us to work with a sub-alphabet $V \subseteq \mathcal{AT}$. We define $SM_V(\varphi) \stackrel{\text{def}}{=} \{\, T \cap V \mid T \in SM(\varphi) \,\}$ as the projection of $SM(\varphi)$ onto some vocabulary V. For any T-support $\mathcal{H}$ and set of atoms V, we write $\mathcal{H}_V$ to stand for $\{H \cap V \mid H \in \mathcal{H}\}$. We say that a T-support $\mathcal{H}$ is *V-feasible* iff there is no $H \subset T$ in $\mathcal{H}$ satisfying that $H \cap V = T \cap V$. The name comes from the fact that, if this condition does not hold for some $\mathcal{H} = [\![\, \varphi \,]\!]^T$ with $H \subset T$, then T cannot be stable for any formula $\varphi \wedge \psi$ with $\psi \in \mathcal{L}_V$ because H and T are indistinguishable for any formula $\psi \in \mathcal{L}_V$.

If F is a fork and $T \subseteq V \subseteq \mathcal{AT}$, we can define the T-view:

$$\langle\!\langle\, F \,\rangle\!\rangle_V^T \stackrel{\text{def}}{=} \downarrow\{\, \mathcal{H}_V \mid \mathcal{H} \in \langle\!\langle\, F \,\rangle\!\rangle^Z \text{ s.t. } Z \cap V = T \text{ and } \mathcal{H} \text{ is } V\text{-feasible} \,\}.$$

Given a fork F, the set $SM_V(F)$ denotes the projection of $SM(F)$ on the set of atoms V, that is $SM_V(F) \stackrel{\text{def}}{=} \{T \cap V \mid T \in SM(F)\}$.

Definition 10 (*Projective Strong Entailment/Equivalence of forks*)
Let F and G be forks and $V \subseteq \mathcal{AT}$ a set of atoms. We say that F *strongly V-entails* G, in symbols $F \mid\!\sim_V G$, if for any fork L in $\mathcal{L}_V$, $SM_V(F \wedge L) \subseteq SM_V(G \wedge L)$. We further say that F and G *are strongly V-equivalent*, in symbols $F \cong_V G$ if both $F \mid\!\sim_V G$ and $G \mid\!\sim_V F$, that is, $SM_V(F \wedge L) = SM_V(G \wedge L)$, for any fork L in $\mathcal{L}_V$.

Note that, when $\mathcal{AT}(F) \cup \mathcal{AT}(G) \subseteq V$, $F \mid\!\sim_V G$ and $F \cong_V G$ respectively amount to the non-projective definitions $F \mid\!\sim G$ and $F \cong G$ we introduced in Definition 2.

Now, the idea of projecting out some atoms of the alphabet can also be applied to explanation graphs, as introduced by Cabalar and Muñiz (2023) (Def. 5) under the name of *node forgetting*.

Definition 11 (*Node forgetting*)
Let $G = \langle\, I, E, \lambda \,\rangle$ be an explanation and let $A \subseteq \mathcal{AT}$ be a set of atoms to be removed. Then, the *node forgetting* operation on G produces the graph $\mathit{forget}(G, A) \stackrel{\text{def}}{=} \langle\, I \setminus A,\ E',\ \lambda \,\rangle$ where E' contains all edges (p_0, p_n) such that there exists a path $(p_0, p_1), (p_1, p_2), \ldots, (p_{n-1}, p_n)$ in E with $n \geq 0$, $\{p_0, p_n\} \cap A = \emptyset$ and $\{p_1, \ldots, p_{n-1}\} \subseteq A$.

Note that E' contains all original edges $(p, q) \in E$ for non-removed atoms $\{p, q\} \cap A = \emptyset$, since they correspond to the case where $n = 1$. It is clear that, if G is acyclic, then $\mathit{forget}(G, A)$ is also acyclic. The only new edges in $\mathit{forget}(G, A)$ are (p_0, p_n) such that there exists a path (p_0, p_1), (p_1, p_2), $\ldots$, (p_{n-1}, p_n) in E. This implies that, if there exist a cycle in $\mathit{forget}(G, A)$, then we will also have a cycle in G.

Appendix B Proofs

Proof of Theorem 2.

Given that $F \mid\!\sim G$ amounts to $\langle\!\langle\, F \,\rangle\!\rangle^T \subseteq \langle\!\langle\, G \,\rangle\!\rangle^T$ for any T, we have to prove:

$$\langle\!\langle\, \varphi \to (\alpha_1 \vee \cdots \vee \alpha_n) \,\rangle\!\rangle^T \ \subseteq\ \langle\!\langle\, \varphi \to (\alpha_1 \mid \cdots \mid \alpha_n) \,\rangle\!\rangle^T$$

for any T. Given the denotation of implication (Def. 1) this amounts to proving $\langle\!\langle\, \alpha_1 \vee \cdots \vee \alpha_n \,\rangle\!\rangle^T \subseteq \langle\!\langle\, \alpha_1 \mid \cdots \mid \alpha_n \,\rangle\!\rangle^T$. Unfolding these two expressions:

$$\begin{array}{rcl} \langle\!\langle\, \alpha_1 \vee \cdots \vee \alpha_n \,\rangle\!\rangle^T & = & \downarrow\![\![\, \alpha_1 \vee \cdots \vee \alpha_n \,]\!]^T = \downarrow\big([\![\, \alpha_1 \,]\!]^T \cup \ldots \cup [\![\, \alpha_n \,]\!]^T\big) \\ \langle\!\langle\, \alpha_1 \mid \cdots \mid \alpha_n \,\rangle\!\rangle^T & = & \langle\!\langle\, \alpha_1 \,\rangle\!\rangle^T \cup \ldots \cup \langle\!\langle\, \alpha_n \,\rangle\!\rangle^T = \downarrow\![\![\, (\alpha_1) \,]\!]^T \cup \ldots \cup \downarrow\![\![\, (\alpha_n) \,]\!]^T \end{array}$$

we essentially have to prove:

$$\downarrow\big([\![\, \alpha_1 \,]\!]^T \cup \ldots \cup [\![\, \alpha_n \,]\!]^T\big) \subseteq \downarrow\![\![\, (\alpha_1) \,]\!]^T \cup \ldots \cup \downarrow\![\![\, (\alpha_n) \,]\!]^T$$

We can distinguish two cases:

1. Suppose $T \models \alpha_i$ for some i, $1 \leq i \leq n$. Then, $T \in [\![\, \alpha_i \,]\!]^T \neq [\,]$ whereas, obviously, $[\![\, \alpha_i \,]\!]^T \subseteq [\![\, \alpha_1 \,]\!]^T \cup \ldots \cup [\![\, \alpha_n \,]\!]^T$ because $1 \leq i \leq n$. As a result, $\big([\![\, \alpha_1 \,]\!]^T \cup \ldots \cup [\![\, \alpha_n \,]\!]^T\big) \preceq [\![\, \alpha_i \,]\!]^T$. This implies:
$$\downarrow\big([\![\, \alpha_1 \,]\!]^T \cup \ldots \cup [\![\, \alpha_n \,]\!]^T\big) \subseteq \downarrow[\![\, \alpha_i \,]\!]^T \subseteq \downarrow\![\![\, (\alpha_1) \,]\!]^T \cup \ldots \cup \downarrow\![\![\, (\alpha_n) \,]\!]^T.$$
2. Suppose $T \not\models \alpha_i$ for all i, $1 \leq i \leq n$. This means $[\![\, \alpha_i \,]\!]^T = [\,]$ for all i. But, then $\downarrow\big([\![\, \alpha_1 \,]\!]^T \cup \ldots \cup [\![\, \alpha_n \,]\!]^T\big) = \downarrow[\,] = \emptyset \subseteq \downarrow\![\![\, (\alpha_1) \,]\!]^T \cup \ldots \cup \downarrow\![\![\, (\alpha_n) \,]\!]^T$ since the empty support $[\,]$ is never included in the ideal so $\downarrow[\,] = \emptyset$.

□

Proof of Theorem 4.

From right to left: $JM(P) \subseteq SM(P^{|})$. Suppose $I = \{a_1, a_2, \ldots, a_n\} \in JM(P)$ and so, $G = \langle\, I, E, \lambda \,\rangle$ is an explanation of I under P. For any $1 \leq i \leq n$, if $\lambda(a_i) = Lb(r_i)$, we know that $a_i \in h(r_i)$ and $I \models Body(r_i)$. We are going to prove that, for any $r \in P$, we can find $\mathcal{H}_r \in \langle\!\langle\, r^{|} \,\rangle\!\rangle^I$ such that:

$$[\, I \,] = \bigcap_{r \in P} \mathcal{H}_r$$

which would mean that $I \in SM(P^{|})$ by applying Definition 1. Notice that $I \models P$, so, for any $r \in P$ such that $I \models Body(r)$, there exists $a_r \in h(r)$ such that $a_r \in I$. Let us define:

$$\mathcal{H}_r := \begin{cases} \overline{[\![\, Body(r) \,]\!]^I} \cup [\![\, a_r \,]\!]^I & \text{if } I \models Body(r) \\ 2^I & \text{otherwise} \end{cases}$$

Notice that $\mathcal{H}_r \in \langle\!\langle\, r^{|} \,\rangle\!\rangle^I$ and suppose that $J \in \mathcal{H}_r$, for any $r \in P$. If $J \neq I$, we get $J \subset I$ (since $\mathcal{H}_r$ is an I-support) and, so there exists $a_0 \in I \setminus J$. Since $a_0 \in I \in JM(P)$, there is a label $\lambda(a_0) = Lb(r_0)$ for some rule r_0 with $a_0 \in h(r_0)$ and $I \models Body(r_0)$. The latter implies $\mathcal{H}_{r_0} = \overline{[\![\, Body(r_0) \,]\!]^I} \cup [\![\, a_0 \,]\!]^I$. Since $J \in \mathcal{H}_{r_0}$ and $a_0 \notin J$ we conclude $\langle\, J, I \,\rangle \not\models Body(r_0)$. Since $I \models Body(r_0)$ then we can find $a_1 \in b^+(r_0) \setminus J$. As a result, there is an edge $(a_1, a_0) \in E$. We can repeat the same reasoning to get a_1 from a_0 and continue like this finding $a_1, a_2, a_3, \ldots$, such that $a_i \in b^+(r_{i-1}) \setminus J$ and getting the edges $(a_i, a_{i-1}) \in E$ for $i \geq 1$. Since I is finite, we know that, for some j, k, $0 \leq j < k \leq |I|$ such that $a_j = a_k$. Notice that we have found a cycle in G given by the path $a_k, a_{k-1}, \ldots, a_j$ which yields a contradiction.

From right to left: $SM(P^{|}) \subseteq JM(P)$. Take $I \in SM(P^{|})$. To prove $I \in JM(P)$, since P is not a labelled program, we start assigning an arbitrary and unique label $Lb(r) := \ell_r$ for every rule $r \in P$. We must then find some explanation $G = \langle\, I, E, \lambda \,\rangle$ of I under (labelled)

P. We will obtain this explanation through another labelled program that results from a particular labelling of $pf(P^{|})$, that is, the translation of $P^{|}$ into a extended disjunctive logic program. According to Def. 3, $pf(P^{|})$ is the result of replacing every rule $r \in P$ with $|h(r)| > 1$ by the rules:

$$x_1 \vee \cdots \vee x_m \quad \leftarrow \quad Body(r) \tag{B1}$$

$$p_i \leftarrow x_i \tag{B2}$$

for each $r \in P$ with $h(r) = \{p_1, \ldots, p_m\}$ and for all $i = 1, \ldots, m$, being x_i new fresh atoms for each rule r. We propose the following labelling for program $pf(P^{|})$:

- If $|h(r)| = 1$ then $Lb(r) = \ell_r$
- Otherwise, $Lb((\text{B1})) := \ell_r^0$, $Lb((\text{B2})) := \ell_r^i$ for all $i = 1, \ldots, m$, where all these labels ℓ_r^0, ℓ_r^i are fresh for each rule r.

Let us call $Aux := \mathcal{AT}(pf(P^{|})) \setminus \mathcal{AT}(P)$, that is, the collection of all fresh variables x_i introduced in $pf(P^{|})$. Since, by Theorem 1, $pf(P^{|}) \cong_{\mathcal{AT}(P^{|})} P^{|}$, we know there exists $\hat{I} \in SM(pf(P^{|}))$ such that $\hat{I} \cap \mathcal{AT}(P^{|}) = I$. Notice that $\hat{I} \in JM(pf(P^{|}))$ by Theorem 3, so there exists a labelled graph $\hat{G} = \langle\, \hat{I}, \hat{E}, \hat{\lambda} \,\rangle$ which is an explanation of $\hat{I}$ under $pf(P^{|})$. Consider the graph $forget(\hat{G}, Aux) = \langle\, (\hat{I} \setminus Aux), E \,\rangle = \langle\, I, E \,\rangle$. In order to complete an explanation of I under P, we need a map $\lambda : I \to Lb(P)$ satisfying the requirements of Definition 5. Take any $p \in I \subseteq \hat{I}$ and its label $\hat{\lambda}(p)$ in $\hat{G}$. We can distinguish two cases:

- $\hat{\lambda}(p)$ is the label of some rule $r \in pf(P^{|})$ that was not modified from P, that is $r \in P$, because $h(r) = \{p\}$. This means $\hat{\lambda}(p) = \ell_r$ and in this case we keep the same label $\lambda(p) := \ell_r$.
- Otherwise, p came from the head of some rule (B2) and so $\hat{\lambda}(p)$ has the form ℓ_r^i for some $i = 1, \ldots, |h(r)|$. Then, we take $\lambda(p) := \ell_r$.

We will proceed to show that the labelled graph formed by $G = \langle\, I, E, \lambda \,\rangle$ is an explanation of I under P. First, notice that, $forget(\hat{G}, Aux)$ is acyclic because $\hat{G}$ was acyclic and node forgetting maintains acyclicity. Second, we observe next that λ is injective. By contradiction, suppose we had two different atoms $p, q \in I$ and $\lambda(p) = \lambda(q)$. We have two cases: if $\lambda(p) = \hat{\lambda}(p)$ and $\lambda(q) = \hat{\lambda}(q)$ then, as $\hat{\lambda}$ is injective, we conclude $p = q$ reaching a contradiction. Otherwise, without loss of generality, suppose $\lambda(p) \neq \hat{\lambda}(p)$ (the proof for q is analogous). Then $\hat{\lambda}(p) = \ell_r^i$ and $\lambda(p) = \ell_r = \lambda(q)$. Therefore, q is also labelled in λ with the same rule r and $|h(r)| > 1$. By construction of λ, this implies $\hat{\lambda}(q) = \ell_r^j$ for some other $j \neq i$, since p and q are different atoms. Now, consider the rules of the form (B2) for the labels we have obtained ℓ_r^i for p and ℓ_r^j for q. But those atoms only occur in the head of one rule of form (B1) that has the label ℓ_r^0. So, we have concluded $\ell_r^0 = \hat{\lambda}(x_i) = \hat{\lambda}(x_j)$ and, since $\hat{\lambda}$ is injective $x_i = x_j$ so $i = j$ reaching a contradiction.

Finally, we have to prove that λ respects the rest of conditions of Definition 5, namely, that for each atom $p \in I$, given the rule $r \in P$ such that $\lambda(p) = Lb(r) = \ell_r$ then: (i) $I \models Body(r)$; and (ii) there is an edge $(q, p) \in E$ for all $q \in b^+(r)$.

For proving (i), we will show that $\hat{I} \models Body(r)$, something that implies $I \models Body(r)$ because $\hat{I} \setminus Aux = I$ and $b(r) \cap Aux = \emptyset$. We have again two cases: when $h(r) = \{p\}$ we just have $\hat{\lambda}(p) = \lambda(p) = \ell_r$ and the rule $r \in P$ also appears unchanged in $pf(P^{|})$ with the same label. Since $\hat{G}$ is an explanation for $\hat{I}$, we conclude $\hat{I} \models Body(r)$. Otherwise $|h(r)| > 1$ and $\hat{\lambda}(p) = \ell_r^i$ for some $i = 1, \ldots, |h(r)|$. But then, we have an edge $(x_i, p) \in \hat{E}$

and $\hat{\lambda}(x_i) = \ell_r^0$ for a rule of the form (B2) for the same $r \in P$. Note that $\hat{I}$ must satisfy the body of (B2) that actually coincides with $Body(r)$, so $\hat{I} \models Body(r)$ again.

For proving (ii), Finally take $q \in b^+(r)$ for some rule r such that $\lambda(p) = Lb(r) = \ell_r$ with $p \in I$. If $h(r) = \{p\}$, then $\hat{\lambda}(p) = \ell_r$, so $(q,p) \in \hat{E}$ and also $(q,p) \in E$ by construction of $forget(\hat{G}, Aux)$ since $\{q,p\} \subseteq \mathcal{AT}(P)$. On the other hand, suppose $\lambda(p) = Lb(r) = \ell_r$ for $r \in P$ with $|h(r)| > 1$. Then $\hat{\lambda}(p) = \ell_r^i$ and we have the labelled rule $\ell_r^i : p \leftarrow x_i$ for some $i = 1, \ldots, |h(r)|$. Therefore $x_i \in \hat{I}$ and $(x_i, p) \in \hat{E}$. Moreover, the only possible label for x_i corresponds to the only rule where it appears $\hat{\lambda}(x_i) = \ell_r^0$. Since the body of that rule is $Body(r)$, we must also have $(q, x_i) \in \hat{E}$ as $q \in b^+(r)$. Finally, since both (q, x_i) and (x_i, p) belong to $\hat{E}$, by construction of $forget(\hat{G}, Aux)$ we conclude $(q,p) \in E$ because $x_i \in Aux$ whereas $p, q \notin Aux$. □

Proof of Theorem 5.

From left to right $CSM(P) \subseteq SM(P^{|})$: suppose that $I \in CSM(P)$, that is, there exists some function sel for which $I \in SM(P^I_{sel})$. We have to prove $I \in SM(P^{|})$ or, if preferred, $[I] \in \langle\!\langle\, P^{|} \,\rangle\!\rangle^I$. Since $P^{|} = \bigwedge_{r \in P} r^{|}$, this amounts to proving that we have some combination of I-supports $\mathcal{H}_r \in \langle\!\langle\, r^{|} \,\rangle\!\rangle^I$ per each $r \in P$ such that $\bigcap_{r \in p} \mathcal{H}_r = [I]$. In particular, for each $r \in P$, take the I-support:

$$\mathcal{H}_r := \begin{cases} \overline{[\![\, Body(r) \,]\!]^I} \cup [\![\, sel(Head(r), I) \,]\!]^I & \text{if } I \models Body(r) \\ 2^I & \text{otherwise} \end{cases}$$

We see next that, according to Definition 1, $\mathcal{H}_r \in \langle\!\langle\, r^{|} \,\rangle\!\rangle^I$. If $I \models Body(r)$, we get $I \models sel(Head(r), I)$ because $I \models P^I_{sel}$, $r^{|} \in P^I_{sel}$ and $r^{|} = (sel(Head(r), I) \leftarrow Body(r))$. Note that, $sel(Head(r), I) \in I$ is some atom in the head of r and so, is one of the branches of the head fork in $r^{|}$. On the other hand, if $I \not\models Body(r)$, $\mathcal{H}_r = 2^I$ and this is trivially included in the ideal $\langle\!\langle\, r^{|} \,\rangle\!\rangle$ as it is the $\preceq$-bottom element. To prove $[I] = \bigcap_{r \in p} \mathcal{H}_r$ we proceed by contradiction: suppose we had a different $J \neq I$, such that $J \in \mathcal{H}_r$ for all $r \in P$. As J belongs to I-supports, we conclude $J \subset I$. But, since I is a stable model of P^I_{sel}, there must be some rule $r' \in P^I_{sel}$ for which $\langle\, J, I \,\rangle \not\models P^I_{sel}$. Suppose r' has the form $sel(Head(r), I) \leftarrow Body(r)$ for one of the rules $r \in P$. Since $I \models r'$ the only possibility is $\langle\, J, I \,\rangle \models Body(r)$ but $\langle\, J, I \,\rangle \not\models sel(Head(r), I)$. This implies both $J \in [\![\, Body(r) \,]\!]^I$ and $J \notin [\![\, sel(Head(r), I) \,]\!]^I$ which is in contradiction with $J \in \mathcal{H}_r$.

From right to left $SM(P^{|}) \subseteq CSM(P)$: take some $I \in SM(P^{|})$. To prove $I \in CSM(P)$ we must find some head selection function sel for I such that $I \in SM(P^I_{sel})$. Consider the translation of $P^{|}$ into a propositional formula $pf(P^{|})$, in our case, the disjunctive logic program we saw in Def. 3. Remember that $pf(P^{|})$ is the result of replacing every rule $r \in P$ with $|h(r)| > 1$ by the rules (B1) and (B2) for each $r \in P$ with $h(r) = \{p_1, \ldots, p_m\}$ and for all $i = 1, \ldots, m$, being x_i new fresh atoms for each rule r. By Theorem 1, there exists some stable model $\hat{I}$ of $pf(P^{|})$ such that $\hat{I} \cap \mathcal{AT} = I$. In fact, it is not hard to see that, for each rule r such that $\hat{I} \models Body(r)$, there exists a *unique* auxiliary atom $x_i \in \hat{I}$ from the head of (B1). This is because, if we had two atoms $x_i, x_j \in \hat{I}$ with $i \neq j$ then the HT interpretation $\langle\, \hat{I} \setminus \{x_j\}, \hat{I} \,\rangle$ would still satisfy $pf(P^{|})$: rule (B1) is still satisfied because $x_i \in \hat{I} \setminus \{x_j\}$ whereas $p_j \leftarrow x_j$ is trivially satisfied since $x_j \notin \hat{I} \setminus \{x_j\}$ whereas $\hat{I} \models (p_j \leftarrow x_j)$. Note also that $\hat{I} \models Body(r)$ iff $I \models Body(r)$ because all atoms occurring

in $Body(r)$ belong to $\mathcal{AT}$. Let us define the following head selection function:

$$sel(Head(r), I) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } h(r) \cap I = \emptyset \\ p_i & \text{if } h(r) \cap I \neq \emptyset, I \models Body(r), \\ & (p_i \leftarrow x_i) \in pf(P^{|}) \text{ and } x_i \in \hat{I} \\ p_j & \text{if } h(r) \cap I \neq \emptyset, I \not\models Body(r), \\ & \text{for some } p_j \in h(r) \cap I \end{cases}$$

We can check that this is a valid selection function, since if selects some head atom among those in $h(r) \cap I$, when this set is not empty. In particular, when $I \models Body(r)$ (and so $\hat{I} \models Body(r)$) we fix the selected atom to p_i for the corresponding auxiliary atom x_i occurring in $\hat{I}$. We know $p_i \in I$ because $p_i \in \hat{I}$ and this atom is not auxiliary, $\hat{I} \cap \mathcal{AT} = I$. On the other hand, when $I \not\models Body(r)$, the selection function is irrelevant, since it is not used in P^I_{sel}. For proving $I \in SM(P^I_{sel})$ we first observe that $I \models P^I_{sel}$ holds by construction of this program because it contains rules of the form $p \leftarrow Body(r)$ where $r \in P$, $I \models Body(r)$ and $p \in h(r) \cap I$. By contradiction, suppose we had some $J \subset I$, $\langle J, I \rangle \models P^I_{sel}$. Then, take:

$$\hat{J} := J \cup \{x_i \in \hat{I} \mid p_i \in J, (p_i \leftarrow x_i) \in pf(P^{|})\}$$

Clearly, $\hat{J} \subset \hat{I}$ and so $\langle \hat{J}, \hat{I} \rangle \not\models pf(P^{|})$ because $\hat{I}$ is a stable model of that program. Then, there must be some rule in $pf(P^{|})$ not satisfied by $\langle \hat{J}, \hat{I} \rangle$ and notice that all rules $r \in P$ with $|h(r)| \leq 1$ are left untouched both in $pf(P^{|})$ and P^I_{sel}, whereas they do not contain auxiliary atoms, so $\langle J, I \rangle \models r$ iff $\langle \hat{J}, \hat{I} \rangle \models r$. In other words, we must have some rule of the form (B1) or (B2) in $pf(P^{|})$ not satisfied by $\langle \hat{J}, \hat{I} \rangle$. Suppose $\langle \hat{J}, \hat{I} \rangle \not\models p_i \leftarrow x_i$ for some of the rules of the form (B2). Then $p_i \notin \hat{J}$ and $x_i \in \hat{J}$ but this is impossible by construction of $\hat{J}$. Suppose, instead, that $\langle \hat{J}, \hat{I} \rangle \not\models x_1 \vee \cdots \vee x_m \leftarrow Body(r)$ for some of the rules of the form (B1). This implies $x_j \notin \hat{J}$ for all $j = 1, \ldots, m$ and $\langle \hat{J}, \hat{I} \rangle \models Body(r)$. Since $Body(r)$ has no auxiliary atoms, the latter implies $\langle J, I \rangle \models Body(r)$. As $\langle J, I \rangle \models sel(Head(r), I) \leftarrow Body(r)$ we conclude $sel(Head(r), I)$ is some atom $p_k \in h(r) \cap J \subset J'$ for some $k = 1, \ldots, m$. But then, by construction of $\hat{J}$ we should have $x_k \in \hat{J}$ for some $k = 1, \ldots, m$ and we reach a contradiction. □

Proof of Theorem 7.

The result follows from Proposition 6 by Shen and Eiter (2019) but making some minor considerations. The application of Shen and Eiter's result has three main differences with respect to what we try to prove: (i) it refers to DI-stable models, which are minimal among the candidate stable models; (ii) it uses well-justified (WJ) semantics to interpret the program P^I_{sel}; (iii) it takes *closed* candidate stable models. Difference (i) is not important, because there exists some minimal candidate stable model iff there exists some candidate stable model (assuming finite interpretations, so we always have a minimal interpretation among a set of interpretations). Namely, the problem of existence of a closed candidate stable model for a disjunctive program using WJ semantics is also NP-complete.

Similarly, difference (ii) is not too relevant either: the specific use of the WJ semantics is important when the program allows arbitrary formulas in the rule bodies, something we do not consider in the current paper. For normal logic programs, WJ collapses to standard stable models. Yet, in the current paper, we do allow P^I_{sel} to be an *extended* normal logic program, that is, we allow double negation in the bodies. However, we can reduce P to

an equivalent program without double negation (modulo auxiliary atoms) while keeping the HT semantics. We define $t_1(P)$ as the result of replacing each doubly negated literal $\neg\neg q$ by $\neg aux$ in all rule bodies and adding the rule $aux \leftarrow \neg q$, where aux is a new fresh atom per each different doubly negated literal. Program $t_1(P)$ is strongly equivalent to program P modulo auxiliary atoms **?** and is obtained by a linear transformation. So the existence of a closed candidate stable model for an *extended* disjunctive logic program is also an NP-complete problem.

Finally, difference (iii) can also be easily overcome: given a disjunctive program P, we can we can obtain its $CSM(P)$ by computing the closed CSMs of another program $t_2(P)$ defined as follows. For each set of rules $r_1, r_2, \ldots, r_n$ sharing the same set of head atoms $Hd(r_1) = Hd(r_2) = \cdots = Hd(r_n)$ replace each rule head of the form $Head(r_i) = p_1 \vee \cdots \vee p_m$ by the disjunction $p_1 \vee \ldots p_m \vee aux_i$ and add the constraint $\bot \leftarrow aux_i$ to the new program, where aux_i is a new, fresh atom per each one of these rules. Program $t_2(P)$ has the same candidate stable models as P (auxiliary atoms are always false) but never repeats the sets of head atoms in disjunctions, so its closed CSMs are just the $CSM(t_2(P))$. Note that transformation $t_2(P)$ is polynomial (we can apply some ordering algorithm on the rules and rule heads to check coincidence of head atoms).

To sum up, deciding $CSM(P) \neq \emptyset$ can be reduced in polynomial time to deciding the existence of a closed candidate stable model of $t_2(t_1(P))$ and the latter is an NP-complete problem according to Proposition 6. □

Proof of Theorem 9.

Let $T \in CSM(P)$. Then, there exists some head selection function sel for P and T such that T is a stable model of P^T_{sel}. To prove $T \in SSM(P)$, we must find a sequence of interpretations $H_0 \subseteq H_1 \subseteq \cdots \subseteq H_n = T$ for some $n \geq 0$ satisfying the conditions in Definition 9. Take the following sequence:

$$\begin{aligned} H_0 &:= \{sel(Head(r), T) \mid r \in P \text{ and } b(r) = \emptyset\}; \\ H_i &:= H_{i-1} \cup \{sel(Head(r), T) \mid r \in P, \langle\, H_{i-1} \cap T, T \,\rangle \models Body(r)\}, \\ &\quad \text{for } i > 0. \end{aligned}$$

We can see that $H_{i-1} \subseteq H_i$ by definition, and so, assuming a finite signature, there exists some $n \geq 0$ such that $H_n = H_{n+1}$. We prove next $\bot \notin H_i \subseteq T$ for all $i \geq 0$ by induction. For the base case, take any $sel(Head(r), T) \in H_0$. Since it is obtained from a rule with empty body $b(r) = \emptyset$, we trivially have $T \models Body(r)$ $(\equiv \top)$ so $sel(Head(r), T)$ becomes a fact in the reduct P^T_{sel}. Since $T \models P^T_{sel}$, we get $sel(Head(r), T) \in T$ and, moreover, $sel(Head(r), T) \neq \bot$ must be some head atom from r. Now, for $i > 0$, note that H_i consists of the union of $H_{i-1} \subseteq T$ (by induction) and $\{sel(Head(r), T) \mid r \in P, \langle\, H_{i-1} \cap T, T \,\rangle \models Body(r)\}$. Take any $sel(Head(r), T)$ in this last set and let r be the rule satisfying $\langle\, H_{i-1} \cap T, T \,\rangle \models Body(r)$. By the Persistence property in HT, $\langle\, H_{i-1} \cap T, T \,\rangle \models Body(r)$ implies $T \models Body(r)$ classically, and so, the rule $sel(Head(r), T) \leftarrow Body(r)$ is included in P^T_{sel}. Again, as $T \models P^T_{sel}$, and $T \models Body(r)$ we conclude $sel(Head(r), T) \in T$ and $sel(Head(r), T) \neq \bot$.

Now, since all $H_i \subseteq T$, we can actually change the condition $\langle H_i \cap T, T \rangle \models \mathit{Body}(r)$ simply by $\langle H_i, T \rangle \models \mathit{Body}(r)$ as in Def. 9:

$$H_i \quad := \quad H_{i-1} \cup \{\mathit{sel}(\mathit{Head}(r), T) \mid r \in P, \langle H_{i-1}, T \rangle \models \mathit{Body}(r)\} \quad \text{for } i > 0$$

Note that these sets so defined $H_0 \subseteq H_1 \subseteq \cdots \subseteq H_n$ satisfy now Conditions 1 and 2 in Def. 9 by construction. Condition 1 holds because all atoms in H_i are selected among the heads $h(r)$ of the same corresponding rules $r \in P$ (remember *sel* never produces $\bot$ in H_i), whereas Condition 2 is satisfied because no other atoms are included in $H_i \setminus H_{i-1}$. However, to obtain $T \in \mathit{SSM}(P)$, we still remain to prove that $H_n = T$ and we only know $H_n \subseteq T$. By contradiction, suppose $H_n \subset T$. Since T is a stable model of P^T_{sel}, we know $\langle H_n, T \rangle \not\models P^T_{sel}$. That is, there is some rule $r \in P$, $r^T_{sel} \in P^T_{sel}$ of the form $r^T_{sel} = (\mathit{sel}(\mathit{Head}(r), T) \leftarrow \mathit{Body}(r))$ such that $\langle H_n, T \rangle \not\models r^T_{sel}$ and $T \models \mathit{Body}(r)$. Since $T \models P^T_{sel}$, the only possibility is $\langle H_n, T \rangle \models \mathit{Body}(r)$ but $\langle H_n, T \rangle \not\models \mathit{sel}(\mathit{Head}(r), T)$. From $\langle H_n, T \rangle \models \mathit{Body}(r)$ and the definition of H_{n+1} we conclude $\mathit{sel}(\mathit{Head}(r), T) \in H_{n+1} = H_n$, reaching a contradiction. □

Proof of Theorem 8.

Suppose that there exists some head selection function *sel* such that $T_{P^I_{sel}}(I) = I$. Let us define a supported graph $G = \langle I, E, \lambda \rangle$ of I under P. For any $p \in I$, there exists $\mathit{Body}(r) \rightarrow \mathit{sel}(\mathit{Head}(r), I) \in P^I_{sel}$ such that $I \models \mathit{Body}(r)$ and $\mathit{sel}(\mathit{Head}(r), I) = p$. We define $\lambda(p) = \mathit{Lb}(r)$. If $\lambda(p) = \mathit{Lb}(r_1) = \lambda(q) = \mathit{Lb}(r_2)$, then $r_1 = r_2$ (different rules have different labels), so $p = \mathit{sel}(\mathit{Head}(r_1), I) = \mathit{sel}(\mathit{Head}(r_2), I) = q$. By definition of λ, if $\lambda(p) = \mathit{Lb}(r)$, then $p = \mathit{sel}(\mathit{Head}(r), I) \in h(r)$ for some rule $r \in P$ such that $I \models \mathit{Body}(r)$. This means that G is a supported graph of I under P.

On the other hand, suppose that $I \in \mathit{SPM}(P)$ being $G = \langle I, E, \lambda \rangle$ a supported graph of I under P. Let us define the following head selection function:

$$\mathit{sel}(\mathit{Head}(r), I) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } h(r) \cap I = \emptyset \\ p & \text{if } h(r) \cap I \neq \emptyset, I \models \mathit{Body}(r), \\ & \text{and } \lambda(p) = \mathit{Lb}(r) \\ q & \text{if } h(r) \cap I \neq \emptyset, I \models \mathit{Body}(r), \\ & \text{for any } q \in h(r) \cap I \text{ such that } \lambda(q) \neq \mathit{Lb}(r) \\ r & \text{if } h(r) \cap I \neq \emptyset, I \not\models \mathit{Body}(r), \\ & \text{for any } r \in h(r) \cap I \end{cases}$$

We have defined $\mathit{sel}(\mathit{Head}(r), I)$ in such a way that, whenever $I \models \mathit{Body}(r)$ and there is an atom $p \in h(r) \cap I$ such that $\lambda(p) = \mathit{Lb}(r)$, we force $\mathit{sel}(\mathit{Head}(r), I) = p$. Notice that, there can not be two different atoms $p, q \in h(r)$ with $\lambda(p) = \lambda(q) = \mathit{Lb}(r)$ because λ is injective. In case there is no such atom $p \in h(r)$ with $\lambda(p) = \mathit{Lb}(r)$, we take another atom $q \in h(r) \cap I$ which always exists because $I \models P$.

We always have that $\mathit{sel}(\mathit{Head}(r), I) \in I$ when $I \models \mathit{Body}(r)$, so $T_{P^I_{sel}}(I) \subseteq I$. Moreover, if $p \in I$, we know that $\lambda(p) = \mathit{Lb}(r_0)$ for some $r_0 \in P$ such that $I \models \mathit{Body}(r_0)$ and $p \in h(r_0)$. The definition of our *sel* function implies that $\mathit{sel}(\mathit{Head}(r_0)), I) = p \in T_{P^I_{sel}}(I)$. □