TPLP **25** (4): 402–416, 2025.

doi:10.1017/S1471068425100185 First published online 26 August 2025

Comparing Non-Minimal Semantics for Disjunction in Answer Set Programming

FELICIDAD AGUADO, PEDRO CABALAR, BRAIS MUÑIZ, GILBERTO PÉREZ and CONCEPCIÓN VIDAL
University of A Coruña, A Coruña, Spain
(*e-mails:* felicidad.aguado@udc.es, cabalar@udc.es, brais.mcastro@udc.es, gperez@udc.es, concepcion.vidalm@udc.es)

submitted 24 July 2025; revised 24 July 2025; accepted 27 July 2025

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 programme (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 programmes, 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 programmes 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 programme P, we first obtain

the so-called *programme reduct* P^I, a programme that corresponds to replacing each negative literal in P by its truth value according to I. Programme P^I amounts to a set of definite Horn clauses and the semantics for these programmes was well-established since the origins of LP. A (consistent) definite programme 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 programme 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 programme 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 programme $P_{(1)}$ 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 programme, the application of $T_{P_{(1)}}$ is undefined and the way to extend it for disjunctive heads is unclear. Stable models for disjunctive programmes 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 programmes to Σ_2^P-complete for disjunctive programmes 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 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 programme coincide with the justified

[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.

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

models of a disjunctive logic programme. 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.[2]

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 programme 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[3] 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:

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

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 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 programme P is a

[2] Appendices with the proofs and some previous additional definitions have been included as supplementary material.

[3] 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.

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

set of rules and, when the programme is finite, we will also understand it as their conjunction. We say that programme P belongs to a syntactic category if all its rules belong to that category.

A *classical interpretation* T is a set of atoms $T \subseteq \mathcal{A}T$. 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 $H \subseteq T \subseteq \mathcal{A}T$; 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:

- $\langle H, T \rangle \not\models \bot$
- $\langle H, T \rangle \models p$ if $p \in H$
- $\langle H, T \rangle \models \varphi \wedge \psi$ if $\langle H, T \rangle \models \varphi$ and $\langle H, T \rangle \models \psi$
- $\langle H, T \rangle \models \varphi \vee \psi$ if$\langle H, T \rangle \models \varphi$ or $\langle H, T \rangle \models \psi$
- $\langle H, T \rangle \models \varphi \rightarrow \psi$ if 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:

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

where φ is a propositional formula over $\mathcal{A}T$ and $p \in \mathcal{A}T$ 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 $(\varphi_1 \mid \ldots \mid \varphi_n)$ – 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 $T \subseteq \mathcal{A}T$, 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

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

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, that is, $^{\downarrow}\Delta = \Delta$. 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:*

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

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 $F \mid\!\sim G$, 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 $F \mid\!\sim G$ and $G \mid\!\sim F$, that is, $SM(F \wedge L) = SM(G \wedge L)$, for any fork L.*

Interestingly, Aguado *et al.* (2019) (Prop. 11) proved that $F \mid\!\sim G$ 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:

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

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

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

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

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 programme P, using less definitions and getting $pf(F)$ in the form of a disjunctive logic programme.

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

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

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

We also define $pf(P^{|}) \stackrel{\text{def}}{=} \bigwedge_{r \in P} pf(r^{|})$. For example, $pf(P^{|}_{(1)})$ 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 programme.* $P^{|}$ *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 programmes with disjunction Gelfond and Lifschitz (1991). This means that disjunction "$\vee$" in logic programmes respects the principle of minimality. For instance, under this definition we still have the same two stable models for programme $P_{(1)}$, 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 programme P as expected.

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

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:

$$\varphi \rightarrow (\alpha_1 \vee \cdots \vee \alpha_n) \mid\sim \varphi \rightarrow (\alpha_1 \mid \cdots \mid \alpha_n)$$

□

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

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

As a result, a disjunctive programme 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 programme $P_{(7)}$ 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:

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

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 programme. 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 programme is said to be *justified* if it admits at least one of these explanation graphs. In the case of normal logic programmes, justified and stable models coincide, but Cabalar and Muñiz (2024) observed that, when the programme is disjunctive, it may have more justified models than stable models. In other words, although every stable model of a disjunctive programme admits an explanation, we may have classical models

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

of the programme 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 programmes 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 programme). *A labelled rule r is an expression of the form ℓ:(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 programme 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 programme, $P^|$ 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 programme P, by $Lb(P)$ we denote the set of labels of the programme $Lb(P) \stackrel{\text{def}}{=} \{Lb(r) \mid r \in P\}$. 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 programme 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 programme 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 programme P by assuming we previously label each rule in P with a unique arbitrary identifier. Note that

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

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 programme, 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 programme 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 – that is models of Clark's completion Clark (1978) – to the disjunctive case. For instance, the programme $P_{(9)}$ 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 *programme completion* defined by Alviano and Dodaro (2016) for disjunctive programmes. 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 programme $P_{(8)}$: 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

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

case, coincide with the stable models of the programme. 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 programmes: 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, 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,[4] 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 programme 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 programme P with respect to some interpretation I and selection function *sel* is defined as the logic programme $P^I_{sel} \stackrel{\text{def}}{=} \{\ sel(Head(r), I) \leftarrow Body(r) \mid I \models Body(r)\ \}$. Note that P^I_{sel} is an extended normal logic programme (possibly containing

[4] 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.

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

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 programme P is said to be a candidate stable model*[5] *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.* □

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

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

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 programme 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 programme 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 programme, under the well-justified semantics Shen et al. (2014), is an NP-complete problem.*

Theorem 7.
Given an extended disjunctive logic programme 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 programmes, 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 programmes as follows.

[5] Or candidate answer set in the original terminology Shen and Eiter (2019).

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

Theorem 8.
Let P be a labelled programme 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[6]). *A model T of an extended disjunctive logic programme P is a strongly supported model of P if there exists a sequence of interpretations $H_0 \subseteq H_1 \subseteq \ldots \subseteq H_n = T$ 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*[7] $\langle H_{i-1}, T \rangle \models Body(r)$.
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. 6 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, $SSM(P_{(1)}) = M(P_{(1)})$ that is 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 programme 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 programme $P_{(9)}$ 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] We use Def. 4 by Doherty *et al.* (2016) but adjusting H_0 as in Def. 11 by Doherty and Szałas (2015).
[7] 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 programmes.

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press

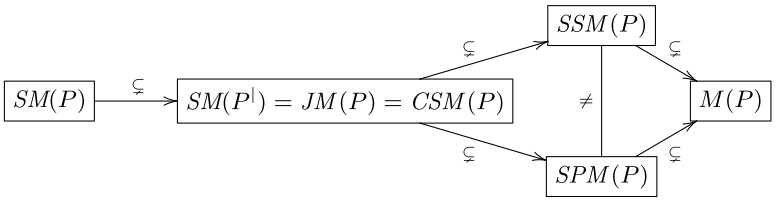

Fig 1. Inclusion relations among several semantics for disjunctive logic programmes.

6 Conclusions

We have studied four different semantics for any disjunctive logic programme P in 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) DI 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 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 $p_1 \vee \ldots \vee p_m \leftarrow Body$ becomes the choice rule $1\{p_1, \ldots, p_m\} \leftarrow Body$ 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.

Supplementary material

The supplementary material for this article can be found at https://doi.org/10.1017/S1471068425100185.

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. 2022. A polynomial reduction of forks into logic programs. *Artificial Intelligence* 308, 103712.

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

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*, Kambhampati, S., Ed. IJCAI/AAAI Press, New York, NY, USA, 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-WS.org, 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, Springer, 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, 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)*, MIT Press, Cambridge, MA, USA, 1070–1080.

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

Heyting, A. 1930. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften. Physikalisch-mathematische Klasse*, 42–56.

Lobo, J., Minker, J. and Rajasekar, A. (1992) Foundations of disjunctive logic programming. In *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, 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)*, Springer, 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, 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.

https://doi.org/10.1017/S1471068425100185 Published online by Cambridge University Press