A Well-Founded Semantics with Disjunction

João Alcântara, Carlos Viegas Damásio and Luís Moniz Pereira

Centro de Inteligência Artificial (CENTRIA)
Departamento de Informática, Universidade Nova de Lisboa
2829-516 Caparica, Portugal. E-mail: jfla|cd|lmp@di.fct.unl.pt

Abstract. In this paper we develop a new semantics for disjunctive logic programs, called *Well-Founded Semantics with Disjunction* (WFS_d), by resorting to a fixed point-based operator. Coinciding with the Well-Founded Semantics (WFS) for normal logic programs, our semantics is uniquely defined for every disjunctive logic program. By exploring examples, we show WFS_d does not agree with any other semantics we have studied, such as Brass and Dix's $D\text{-}WFS$, Przymusinski's Static, Baral *et al*'s $GDWFS$, Wang's $WFDS$, and van Gelder *et al*'s $SWFS$. Despite that, we ensure WFS_d is strictly stronger than $D\text{-}WFS$ by guaranteing WFS_d allows the five, desirable, program transformations proposed by Brass and Dix: unfolding, elimination of tautologies and non-minimal rules, and positive and negative reduction.

1 Introduction

To be, or not to be: that is the question
William Shakespeare in Hamlet (III, I)

Not just Hamlet has been involved in disquisitions on disjunctions; since the early eighties the Logic Programming Community too has devoted great effort to provide reasonable answers to disjunctive forms of reasoning. The pioneer work in the field is [6], where Minker presents the *Generalized Closed World Assumption (GCWA)* to define model-theoretically the default negation in disjunctive theories. For Horn theories [12], $GCWA$ reduces to the *Closed World Assumption*.

For a Horn theory extended to allow disjunction in the head of its rules, named positive-disjunctive logic program ($PDLP$), we can easily match its semantics to the set of *minimal Herbrand models* or to the *minimal model state*, which is the collection of all positive disjunctions holding in every minimal Herbrand Model. Both paths can equivalently yield the expected meaning. In [7], Minker and Rajasekar defined the operator $\mathcal{T}_P^s$, whose least fixed point represents the minimal model state of a $PDLP$. Regarding the path based on minimal Herbrand models, in [3], for $PDLP$s without function symbols, Fernández and Minker defined the operator $\mathcal{T}_P^M$ on sets of Herbrand interpretations whose least fixed point is the set of minimal Herbrand models of the program. Afterwards, in [14], the authors extended $\mathcal{T}_P^M$ in terms of $\mathcal{T}_P^{INT}$ in order to deal also with function symbols in such programs.

Whereas the minimal Herbrand models (or minimal model state) are largely accepted as capturing the meaning of *PDLP*s, the same cannot be said for *disjunctive logic programs*, i.e. normal logic programs whose syntax allows disjunction in the head of their rules. Indeed, as well as for normal logic programs, there are two major approaches when considering the semantics of disjunctive logic programs: one extends stable models semantics [4], and the other, more skeptical, extends well-founded models semantics (*WFS*) [18].

Unlike stable model semantics, *WFS* is defined for every normal logic program. Indeed, for each there is a unique *WFM* intended to capture its meaning or declarative semantics. This is attained by abandoning classical two-valued interpretations and permitting partial interpretations, wherein a third value is introduced to represent undefinedness. The result is an elegant semantics, advantageously compared, from a computational viewpoint, to stable models.

When moving to disjunctive logic programming one would hope that such properties be preserved. Notwithstanding, given the numerous proposals (for a non-exhaustive list, see [13, 1, 9, 2, 19]), we would agree that a well-founded semantics for disjunctive logic programs representing their intended meaning is not consensual. The issues arise from the start in identifying which meaning someone assigns to disjunction in logic programming: should it be exclusive? inclusive? or should it be the so-called epistemic disjunction, designed to be exclusive as far as possible? This barrier surmounted, which properties should endorse the relation between disjunction and default negation? Such concerns are not only formidable, but they involve aspects so diverse that even the choice of a touchstone to appraise disjunctive well-founded semantics is hardly achievable.

Prospectively, we propound that a good well-founded semantics for disjunctive logic programs should, at least:

- Coincide with *WFS* on normal logic programs.
- Be uniquely defined for every disjunctive logic program.
- Be definable by both a fixed point operator and a model-theoretically one.
- Comply with the five program transformations proposed by Brass and Dix: unfolding, elimination of tautologies and non-minimal rules, positive and negative reduction. In other words, their *D-WFS* should be the weakest well-founded semantics for disjunctive logic programs.
- Allow further extensions to assimilate explicit negation, both in its explosive as well as in its paraconsistent varieties.

In our studies about the applicability of logic programming as a suitable tool to represent knowledge and reasoning we have observed that by generalising to a set of interpretations the Ω_P operator [8], used to define *WFS*, and by positing a new domain to deal specifically with default negation, we are able to define a new well-founded semantics for disjunctive logic programs. This semantics, dubbed *Well-Founded Semantics with Disjunction* (WFS_d), shows to be worthy enough and to satisfy the conditions just enrolled.

The remainder of this work is spent as follows. First, by recalling in Section 2 how to determine the minimal models of a *PDLP* via the $\mathcal{T}_P^{INT}$ operator [14]. We then reserve Section 3 to present our main contribution: the definition of WFS_d established by a fixed point operator. Section 4 is devoted to guarantee, by means of examples, that WFS_d generally does not agree with any of the semantics for disjunctive logic programs we have studied. Going into more detail, we then show that WFS_d is strictly stronger than $D\text{-}WFS$ [2]. The paper terminates with conclusions and future work.

2 Minimal models of positive-disjunctive logic programs

In this section we recall [14] in showing how to capture the minimal models of a *PDLP* via the fixed point operator $\mathcal{T}_P^{INT}$. As the authors emphasise, although non-continuous it converges to its least fixed point in at most ω iterations. We also present basic concepts employed throughout this paper.

Let us now present disjunctive (and positive-disjunctive) logic programs in more formal terms. Given a first order language $\mathfrak{L}$, a disjunctive logic program is a set of logical inference rules r of the form

$$r = A_1 \vee \ldots \vee A_l \leftarrow B_1 \wedge \ldots \wedge B_m \wedge not\ C_1 \ldots \wedge not\ C_n \tag{1}$$

in which the A_i, B_j, C_k are atoms in $\mathfrak{L}$, and $l, m, n \in \mathbb{N}$. The set of all *ground instances* of the rules in P is denoted by $gnd(P)$. If $m = n = 0$ the rule is said a *fact*, and the symbol $\leftarrow$ may be dropped.

In particular, P is defined as a positive-disjunctive logic program (*PDLP*) if $\forall r \in P$ we have that $k = 0$, i.e. P is deprived of default negation.

By *Herbrand base* H_{B_P} *of a disjunctive logic program* P, we mean the set of all ground atoms over the language of P. In order to study the meaning of these programs, the notion of interpretation is introduced:

Definition 1 (Interpretations and coins). *Given a program P, we define the following terms:*

- *By* Herbrand interpretation *(or just interpretation) we mean a set $I \subseteq H_{B_P}$. The set of all interpretations is denoted by $\mathcal{H}_{\mathcal{I}_P}$.*
- *For conciseness, by* coin *we mean a collection of interpretations. In particular, $\mathcal{H}_{\mathcal{I}_P}$ is a coin.*

When dealing with a *coin* $\mathcal{I}$, we will frequently refer to the operators *min*, *exp*, and *imp* for $\mathcal{I}$:

$min(\mathcal{I}) = \{I \in \mathcal{I} \mid \nexists J \in \mathcal{I} : J \subset I\}$,
$exp(\mathcal{I}) = \{I \in \mathcal{H}_{\mathcal{I}_\mathcal{P}} \mid \exists J \in \mathcal{I} : J \subseteq I\}$,
$imp(\mathcal{I}) = \{I \in \mathcal{H}_{\mathcal{I}_\mathcal{P}} \mid \exists J \in \mathcal{I} : I \subseteq J\}$.

A coin $\mathcal{I}$ is called *expanded* if, for each interpretation $I \in \mathcal{I}$, $\mathcal{I}$ contains all interpretations that are supersets of I, i.e. $\mathcal{I} = exp(\mathcal{I})$. Similarly, $\mathcal{I}$ is called *self-contained* if, for each interpretation $I \in \mathcal{I}$, $\mathcal{I}$ contains all interpretations that are subsets of I, i.e. $\mathcal{I} = imp(\mathcal{I})$. In order to illustrate applications of these operators, let us choose the Herbrand base $H_{B_P} = \{a, b, c\}$ and the coin $\mathcal{I} = \{\{a\}, \{a, b\}, \{b, c\}\}$; we have:

$min(\mathcal{I}) = \{\{a\}, \{b, c\}\}$,
$exp(\mathcal{I}) = \mathcal{I} \cup \{\{a, c\}, \{a, b, c\}\}$,
$imp(\mathcal{I}) = \mathcal{I} \cup \{\{\}, \{b\}, \{c\}\}$.

For *PDLP*s, we can distinguish the models as per below:

Definition 2 (Satisfaction and model). *An interpretation I satisfies a rule $r = A_1 \vee \ldots \vee A_l \leftarrow B_1, \ldots, B_m$ of a PDLP P iff it holds that $\{B_i \mid 1 \leq i \leq m\} \subseteq I$ implies that there exists $A_j \in I$ such that $1 \leq j \leq l$. I is a model of P iff for every $r \in P$, I satisfies r. The set of all models of P is denoted by $\mathcal{M}_{\mathcal{OD}}(P)$.*

We say a model M of P is *minimal* if there is no model M' for which $M' \subset M$. In order to reason with *PDLP*s, we consider the set $\mathcal{MM}_P$ of all minimal models of P. For Horn theories, the unique minimal (least) model can be obtained by iteratively applying an immediate consequences operator introduced by van Emden and Kowalski [17]. This operator can be generalised to deal with *PDLP*s:

Definition 3 (T_P mapping). *Let P be a PDLP and I be an interpretation. Then the immediate consequences mapping $T_P : \mathcal{H}_{\mathcal{I}_P} \rightarrow 2^{\mathcal{H}_{\mathcal{I}_P}}$ is defined as follows:*

$T_P(I) = \{J \mid J \cap \{a_1, \ldots, a_l\} \neq \emptyset$ for every $r = a_1 \vee \ldots \vee a_l \leftarrow b_1, \ldots, b_m \in P$ such that $\{b_1, \ldots, b_m\} \subseteq I, I \subseteq J\}$.

Thus, from each single interpretation, various interpretations may be derived through the T_P mapping. Then the $\mathcal{T}_P^{INT}$ operator can be defined by applying T_P to each interpretation of a coin. Formally:

Definition 4 ($\mathcal{T}_P^{INT}$ Operator). *Let P be a PDLP and $\mathcal{I}$ be a coin. We define $\mathcal{T}_P^{INT} : 2^{\mathcal{H}_{\mathcal{I}_P}} \rightarrow 2^{\mathcal{H}_{\mathcal{I}_P}}$ such that*

$$\mathcal{T}_P^{INT}(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} T_P(I)$$

In their studies to define an immediate consequences operator for *PDLP*s, Fernández and Minker [3] explored the subsumption relation $\sqsubseteq$ established for coins $\mathcal{I}, \mathcal{J} \in 2^{\mathcal{H}_{\mathcal{I}_P}}$:

$$\mathcal{I} \sqsubseteq \mathcal{J} \textit{ iff } \forall J \in \mathcal{J} : \exists I \in \mathcal{I} : I \subseteq J$$

In fact, $\sqsubseteq$ corresponds to the Hoare ordering (see [15]) associated with the partially ordered set $(\mathcal{H}_{\mathcal{I}_P}, \subseteq)$. For example, letting $\mathcal{I} = \{\{a\}, \{c\}\}$ and $\mathcal{J} = \{\{a\}, \{a, b\}, \{b, c\}\}$, we have $\mathcal{I} \sqsubseteq \mathcal{J}$.

They also define an equivalence class with respect to $\sqsubseteq$, denoted by $\equiv_{\sqsubseteq}$, when coins subsume one another, i.e.

$$\mathcal{I} \equiv_{\sqsubseteq} \mathcal{J} \textit{ iff } \mathcal{I} \sqsubseteq \mathcal{J} \textit{ and } \mathcal{J} \sqsubseteq \mathcal{I}$$

Nonetheless, as it stands, the characterisation of $\mathcal{T}_P^{INT}$ may present some problems. The reason is that $\sqsubseteq$ on $2^{\mathcal{H}_{\mathcal{I}_P}}$ is reflexive and transitive, but not anti-symmetric; the pair $\mathcal{O} =< 2^{\mathcal{H}_{\mathcal{I}_P}}, \sqsubseteq>$ is not a (complete) lattice. This jeopardises making use of the results supported by the Knaster-Tarski theorem [16].

In order to overcome this, extending Fernández and Minker's work [3], Seipel *et al* [14] proposed the lattice $\mathcal{O}_{exp} =< 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{exp}, \sqsubseteq>$, in which $2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{exp}$ corresponds to the set of all expanded coins. In $\mathcal{O}_{exp}$ the relation $\sqsubseteq$ reduces to superset inclusion, i.e. for every $\mathcal{I}, \mathcal{J} \in 2^{\mathcal{H}_{\mathcal{I}_P}}$ it holds that

$$\mathcal{I} \sqsubseteq \mathcal{J} \ \textit{iff} \ exp(\mathcal{J}) \subseteq exp(\mathcal{I}) \tag{2}$$

This suffices to ensure $\mathcal{O}_{exp}$ is *complete* and that the least upper bound and the greatest lower bound of a set $X \subseteq 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{exp}$ of expanded coins are respectively given by

$$lub_{exp}(X) = \bigcap_{\mathcal{I} \in X} \mathcal{I}, \qquad glb_{exp}(X) = \bigcup_{\mathcal{I} \in X} \mathcal{I}$$

The bottom element of $\mathcal{O}_{exp}$ is the coin $\perp_{exp} = \mathcal{H}_{\mathcal{I}_P}$. Notice that both the union and the intersection of a set X of expanded interpretations is an expanded interpretation.

The $\mathcal{T}_P^{INT}$ operator is guaranteed monotonic with respect to $\sqsubseteq$, i.e. if $\mathcal{I}_1 \sqsubseteq \mathcal{I}_2$ then $\mathcal{T}_P^{INT}(\mathcal{I}_1) \sqsubseteq \mathcal{T}_P^{INT}(\mathcal{I}_2)$, so, according to the Knaster-Tarski theorem, a least fixed point of $\mathcal{T}_P^{INT}$ on $< 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{exp}, \sqsubseteq>$ is guaranteed to exist. Hence we can employ the *ordinal powers* of $\mathcal{T}_P^{INT}$ defined below to reach this least fixed point:

Definition 5 (Ordinal powers of $\mathcal{T}_P^{INT}$). *Let P be a PDLP.*

1. *For $\mathcal{I} \in 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{exp}$, the ordinal powers $\mathcal{T}_P^{INT\ \uparrow\alpha}$ are defined by*

   $\mathcal{T}_P^{INT\ \uparrow 0} = \perp_{exp} = \mathcal{H}_{\mathcal{I}_P}$.

   $\mathcal{T}_P^{INT\ \uparrow\alpha} = \mathcal{T}_P^{INT}(\mathcal{T}_P^{INT\ \uparrow\alpha-1})$, *for successor ordinal α.*

   $\mathcal{T}_P^{INT\ \uparrow\alpha} = lub_{exp}(\{\mathcal{T}_P^{INT\ \uparrow\beta} \mid \beta < \alpha\}) = \bigcap_{\beta<\alpha} \mathcal{T}_P^{INT\ \uparrow\beta}$, *for limit ordinal α.*

2. *The least fixed point of $\mathcal{T}_P^{INT}$, denoted by $\mathcal{T}_P^{coin}$, is given by $\mathcal{T}_P^{INT\ \uparrow\alpha}$, in which α is the smallest ordinal such that $\mathcal{T}_P^{INT\ \uparrow\alpha} = \mathcal{T}_P^{INT\ \uparrow\alpha+1}$.*

The authors show in [14] that even though $\mathcal{T}_P^{INT}$ is *not continuous*, it reaches its least fixed point in at most ω iterations. They also guarantee that $\mathcal{T}_P^{INT}$ generalises the characterisation of the minimal model semantics that Fernández and Minker obtained for disjunctive deductive databases to arbitrary *PDLP*s, namely that

$$\mathcal{MM}_P \equiv_{\sqsubseteq} exp(\mathcal{MM}_P) = \mathcal{T}_P^{coin} = \mathcal{T}_P^{INT\ \uparrow\omega} \tag{3}$$

In the next section, we will investigate how to explore the results exhibited here in order to define a well-founded semantics for disjunctive logic programs.

3 A fixed point based definition of $\boldsymbol{WFS}_d$

The definition of an adequate semantics for default negation in disjunctive logic programs has puzzled many researchers [10, 13, 1, 11, 9, 2, 19] employed to define well-founded semantics [8] to a set of interpretations, we propose a new disjunctive version of the well-founded semantics. A fixed point-based definition is then presented to characterise it.

In normal logic programs, well-founded semantics may be defined by introducing partial interpretations, which are a pair of interpretations $I^p =< I^t, I^{tu} >$; the I^t component is designed to determine what is true in I^p, whilst I^{tu} what is "non-false", i.e. true or undefined, in I^p. As is suggestive, I^t is employed to evaluate atoms, and I^{tu} default negated atoms, the later being true if and only if their atoms are not in I^{tu}.

Our idea is to exploit this notion of pair of interpretations to characterise well-founded semantics for disjunctive logic programs, but now considering the pair $\mathcal{I}^p =< \mathcal{I}^t, \mathcal{I}^{tu} >$, in which both $\mathcal{I}^t$ and $\mathcal{I}^{tu}$ are coins. Pairs of coins will be used to evaluate disjunctive clauses as follows:

A *disjunctive ground clause* $a_1 \vee \ldots \vee a_m \vee$ *not* $b_1 \vee \ldots \vee$ *not* b_n is true in $\mathcal{I}^p =< \mathcal{I}^t, \mathcal{I}^{tu} >$ iff

D_1 For each $I_1 \in \mathcal{I}^t$, $\exists a_i \in \{a_1, \ldots, a_m\}$, such that $a_i \in I_1$
or
D_2 For each $I_2 \in \mathcal{I}^{tu}$, $\exists b_j \in \{b_1, \ldots, b_n\}$, such that $b_j \notin I_2$.

As we shall unveil, conditions D_1 and D_2 generalise the evaluation mechanism of positive-disjunctive and negative-disjunctive ground clauses in *PDLP*s. Condition D_1 is motivated by the way positive-disjunctive ground clauses are characterised in terms of minimal models of *PDLP*, as Minker proved [6]:

Theorem 1. *Let P be a PDLP. A positive-disjunctive ground clause $a_1 \vee \ldots \vee a_n$ is a logical consequence of P if and only if $a_1 \vee \ldots \vee a_n$ is* true *in every minimal model of P.*

Then we have the following result, linking condition D_1 to the notion of logical consequence of a positive-disjunctive logic program:

Proposition 1. *Let P be a PDLP, $a_1 \vee \ldots \vee a_n$ be a positive-disjunctive ground clause such that $a_i \in H_{B_P}$, $1 \leq i \leq n$, and $exp(\mathcal{M}^t) = exp(\mathcal{MM}_P)$[1]. We have*

$$a_1 \vee \ldots \vee a_n \textit{ is a logical consequence of } P$$

iff

$$a_1 \vee \ldots \vee a_n \textit{ is true in } \mathcal{M}^p =< \mathcal{M}^t, \mathcal{M}^{tu} >$$

in which $\mathcal{M}^{tu}$ is any set of interpretations.

Regarding D_2, it provides a mechanism to evaluate negative-disjunctive ground clauses based on a principle known as Extended Generalised Closed-World Assumption (*EGCWA*), that we recall below:

[1] Note that the theorem holds for $\mathcal{M}^t = \mathcal{MM}_P$.

Definition 6 (Semantic definition of EGCWA). *[5] Let P be a PDLP and $\mathcal{C}$ be a negative-disjunctive ground clause not $a_1 \vee \ldots \vee$ not a_n such that $a_i \in H_{B_P}$, $1 \leq i \leq n$. Then the* model-theoretic definition of *EGCWA is given by*

$$EGCWA(P) = \{\mathcal{C} \mid \mathcal{C} \text{ is true in all minimal models of } P\}.$$

The relation between *EGCWA* and how to evaluate negative-disjunctive ground clauses is described by Proposition 2:

Proposition 2. *Let P be a PDLP, $\mathcal{C}$ be a negative-disjunctive ground clause not $A_1 \vee \ldots \vee$ not A_n such that $A_i \in H_{B_P}$, $1 \leq i \leq n$, $imp(\mathcal{M}^{tu}) = imp(\mathcal{MM}_P)$, and $\mathcal{MM}_P$ is the set of minimal models of P. We have*

$$not\ A_1 \vee \ldots \vee not\ A_n \in EGCWA(P)$$

iff

$$not\ A_1 \vee \ldots \vee not\ A_n \text{ is true in } \mathcal{M}^p =< \mathcal{M}^t, \mathcal{M}^{tu} >$$

in which $\mathcal{M}^t$ is any coin.

Proposition 2 ensures that D_2 corresponds to evaluating negative-disjunctive ground clauses according to *EGCWA*. It emphasises as well, like in Proposition 1, the role played by minimal models in determining what follows from a *PDLP*. So, if we wish to provide meaning for the more complex syntax of disjunctive logic programs, determining them is fundamental.

As expounded in Section 2, the ordinal powers for the $\mathcal{T}_P^{INT}$ operator on the lattice $\mathcal{O}_{exp} =< 2_{exp}^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}, \sqsubseteq>$ are suitable to express what is true in a disjunctive positive logic program. Ideally, a similar mathematical equipment would give us what is false by default in a *PDLP*. The problem is that the relation $\sqsubseteq$ does not preserve the notion of falsity by default, as shown below:

Example 1. Let $\mathcal{I}_1^p =< \mathcal{I}_1^t, \mathcal{I}_1^{tu} >$ and $\mathcal{I}_2^p =< \mathcal{I}_2^t, \mathcal{I}_2^{tu} >$, in which $\mathcal{I}_1^{tu} = \{\{a\}\}$, $\mathcal{I}_2^{tu} = \{\{a\}, \{a, b\}\}$, and $\mathcal{I}_1^t$ and $\mathcal{I}_2^t$ are arbitrary coins.

If $\sqsubseteq$ was suitable to deal with default negation, equivalent elements of $2^{\mathcal{H}_{\mathcal{I}_P}}$ according to $\sqsubseteq$ would correspond to equivalent evaluations for default negated atoms. That is not the case in Example 1, wherein $\mathcal{I}_1^{tu} \equiv_{\sqsubseteq} \mathcal{I}_2^{tu}$, but intuitively b is false by default in $\mathcal{I}_1^p$, whilst it is not false by default in $\mathcal{I}_2^p$.

In conclusion, the notion of equivalence suggested by $\sqsubseteq$ does not correspond generally to equivalent evaluations for default negated atoms, we thence introduce the new relation $\preceq$, which corresponds to the Smyth ordering [15] associated with the partially ordered set $(\mathcal{H}_{\mathcal{I}_P}, \subseteq)$ amongst coins:

$$\mathcal{I} \preceq \mathcal{J} \text{ iff } \forall I \in \mathcal{I}, \exists J \in \mathcal{J} : I \subseteq J \tag{4}$$

The relation $\preceq$ works as the dual of $\sqsubseteq$. Like for $\sqsubseteq$, the relation $\preceq$ on $2^{\mathcal{H}_{\mathcal{I}_P}}$ does not represent a partial ordering because the anti-symmetry property does not hold. In any case, considering the domain $2_{imp}^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}$, defined as the set of all coins $\mathcal{I} \in 2^{\mathcal{H}_{\mathcal{I}_P}}$ such that $imp(\mathcal{I}) = \mathcal{I}$, we know that $\mathcal{O}_{min} =< 2_{imp}^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}, \preceq>$ is a complete lattice. Let us start by guaranteeing the following result:

Theorem 2. *For every $\mathcal{I}, \mathcal{J} \in 2^{\mathcal{H}_{\mathcal{I}_P}}$ it holds that $\mathcal{I} \preceq \mathcal{J}$ iff $imp(\mathcal{I}) \subseteq imp(\mathcal{J})$.*

Hence if $\mathcal{I}, \mathcal{J} \in 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{imp}$ we have $\mathcal{I} \preceq \mathcal{J}$ iff $I \subseteq J$. This is enough to guarantee that $\mathcal{O}_{imp} =< 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{imp}, \preceq>$ is a complete lattice in which $\preceq$ reduces to subset inclusion, whereas the least upper bound and the greatest lower bound of a set $X \subseteq 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{imp}$ are given respectively by

$$lub_{imp}(X) = \bigcup_{\mathcal{I} \in X} \mathcal{I}, \qquad glb_{imp}(X) = \bigcap_{\mathcal{I} \in X} \mathcal{I}$$

The bottom element of $\mathcal{O}_{imp}$ is the coin $\bot_{imp} = \{\{\}\}$, and the top one is $\top_{imp} = \mathcal{H}_{\mathcal{I}_P}$. Notice that both the union and the intersection of a set X of coins $\mathcal{I} \in 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{imp}$ is also an element of $2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{imp}$.

We are going to show in the sequel that the lattice $\mathcal{O}_{imp}$ complies with our objective of providing a suitable means to interpret default negation in a disjunctive logic program. For the moment, let us reconsider Example 1:

Example 2. Let $\mathcal{O}_{imp} =< \{\{a, b\}, \{a\}, \{b\}, \{\}\}, \preceq>$, and both $\mathcal{I}^{tu}_1 = \{\{a\}\}$ and $\mathcal{I}^{tu}_2 = \{\{a\}, \{a, b\}\}$ be coins.

As we can see, $\mathcal{I}^{tu}_1$ and $\mathcal{I}^{tu}_2$ are not equivalent according to $\preceq$, reflecting that b is false by default in $\mathcal{I}^p_1$, whilst it is not false by default in $\mathcal{I}^p_2$.

We prove now that in $2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{imp}$ the truth-value assigned to a negative-disjunctive clause is preserved in the equivalence class created by $\preceq$:

Proposition 3. *If a negative-disjunctive clause $\mathcal{C} = not\ A_1 \vee \ldots \vee not\ A_n$ is true in the pair of coins $\mathcal{I}^p =< \mathcal{I}^t, \mathcal{I}^{tu} >$, in which $\mathcal{I}^{tu} \in \mathcal{H}_{\mathcal{I}_P}$, then $\mathcal{C}$ is true in any pair of coins $\mathcal{J}^p =< \mathcal{J}^t, \mathcal{J}^{tu} >$ such that $imp(\mathcal{J}^{tu}) = imp(\mathcal{I}^{tu})$.*

Because in $2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{imp}$ the equivalence classes preserve the truth-value assigned to negative-disjunctive clauses, then, instead of using the domain $\mathcal{H}_{\mathcal{I}_\mathcal{P}}$ the subdomain $2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{imp}$ rather would be suitable to provide semantics for negative-disjunctive clauses in disjunctive logic programs, whereas $2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{exp}$ would be suitable to deal with positive-disjunctive clauses. With these ideas in mind, we define partial coins as follows:

Definition 7 (Partial coins). *A partial coin is a pair $\mathcal{I}^p =< \mathcal{I}^t, \mathcal{I}^{tu} >$ such that $\mathcal{I}^t \in 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{exp}$ and $\mathcal{I}^{tu} \in 2^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}_{imp}$.*

Accordingly, partial coins are designed to evaluate positive and negative-disjunctive rules, and can be ordered in the following way:

Definition 8 (Truth and knowledge ordering). *Let $\mathcal{I}^p_1$ and $\mathcal{I}^p_2$ be two partial coins. The truth and knowledge orderings among them are defined by*

Truth ordering: $\mathcal{I}^p_1 \sqsubseteq_t \mathcal{I}^p_2$ *iff* $\mathcal{I}^t_1 \sqsubseteq \mathcal{I}^t_2$ *and* $\mathcal{I}^{tu}_1 \preceq \mathcal{I}^{tu}_2$.

Knowledge ordering: $\mathcal{I}^p_1 \sqsubseteq_k \mathcal{I}^p_2$ *iff* $\mathcal{I}^t_1 \sqsubseteq \mathcal{I}^t_2$ *and* $\mathcal{I}^{tu}_2 \preceq \mathcal{I}^{tu}_1$.

Associated with partial coins, we propose the following notions of models:

Definition 9 (T and TU-models). *Let P be a PDLP, and $\mathcal{M}^p = <\mathcal{M}^t, \mathcal{M}^{tu}>$ be a partial coin. We say that*

- *$\mathcal{M}^t$ is a T-model of P iff, for every $M \in \mathcal{M}^t$, M is a model of P.*
- *$\mathcal{M}^{tu}$ is a TU-model of P iff, for every $M \in \mathcal{M}^{tu}$, there exists a model $M' \in \mathcal{M}^{tu}$ of P such that $M \subseteq M'$.*

We refer to $\mathcal{M}^t$ as minimal T-model of P whenever there is no T-model $\mathcal{M}$ of P such that $\mathcal{M} \sqsubset \mathcal{M}^t$. Similarly, we refer to $\mathcal{M}^{tu}$ as minimal TU-model of P whenever there is no TU-model $\mathcal{M}'$ of P such that $\mathcal{M}' \prec \mathcal{M}^{tu}$.

In point of fact, we saw in Section 2 that, for each *PDLP* P, there is a unique minimal (least) T-model $\mathcal{MM}_P \in \mathcal{M}_{\mathcal{OD}}(P)$ [14] captured by the $\mathcal{T}_P^{INT}$ operator, i.e. $\mathcal{MM}_P \equiv_{\sqsubseteq} \mathcal{T}_P^{coin} = \mathcal{T}_P^{INT\ \uparrow\omega}$.

Before continuing the study of TU-models, we introduce the $\mathcal{TU}_P^{coin}$ mapping:

Definition 10 ($\mathcal{TU}_P^{coin}$ mapping). *Let P be a PDLP. We define*

$$\mathcal{TU}_P^{coin} = imp(\mathcal{MM}_P) = imp(min(\mathcal{T}_P^{INT\ \uparrow\omega}))$$

The $\mathcal{TU}_P^{coin}$ mapping will be used to obtain the set of interpretations in $2_{imp}^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}$ corresponding to the minimal models of a *PDLP*. Concerning its meaning, unlike for the $\mathcal{T}_P^{INT}$ operator, from which one can obtain the least T-model of a positive-disjunctive definite logic program P (see Equation (3)), one cannot draw out the least TU-model of P, under $\preceq$, with the operator $\mathcal{TU}_P^{coin}$.

Example 3. Let $P = \{a \vee b\}$ be a *PDLP*. Although $\mathcal{TU}_P^{coin} = \{\{a\}, \{b\}, \{\}\}$, P has two minimal TU-models: $\{\{a\}, \{\}\}$ *and* $\{\{b\}, \{\}\}$.

Notwithstanding, as we are about to prove, $\mathcal{TU}_P^{coin}$ corresponds to the least upper bound in $2_{imp}^{\mathcal{H}_{\mathcal{I}_\mathcal{P}}}$ of the minimal *TU*-models of a *PDLP* P:

Theorem 3. *Let P be a PDLP, and $\mathcal{M} = \bigcup\{\mathcal{M}^{tu} \mid \mathcal{M}^{tu}$ is a minimal TU-model of $P\}$. We have that $\mathcal{M} = \mathcal{TU}_P^{coin}$.*

Even though $\mathcal{T}_P^{coin}$ and $\mathcal{TU}_P^{coin}$ mappings were defined in different domains in order to attain different objectives, they do not disagree with each other in what concerns the set of minimal models of a *PDLP*:

Theorem 4. *Let P be a PDLP. We have*

$$\mathcal{T}_P^{coin} \cap \mathcal{TU}_P^{coin} = \mathcal{MM}_P(P).$$

The $\mathcal{T}_P^{coin}$ and $\mathcal{TU}_P^{coin}$ mappings therefore share the core meaning of a *PDLP* P represented by its sets of minimal models. The difference in these mappings is focused on the way that such minimal models are interpreted. In order to capture what is true in P the sets of minimal models of P are closed upward (expanded) in $\mathcal{T}_P^{coin}$, whereas this very set is closed downward in $\mathcal{TU}_P^{coin}$ to capture what is non-false in P.

Example 4. Let $P = \{a \vee b, b \vee c\}$ be a *PDLP*. We have

$$\mathcal{M}_{\mathcal{OD}}(P) = \{\{b\}, \{a, c\}, \{a, b\}, \{b, c\}, \{a, b, c\}\}$$

$$\mathcal{MM}_P(P) = \{\{b\}, \{a, c\}\}$$

and

$$\mathcal{T}_P^{INT} \uparrow^0 = \{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}\{b, c\}, \{a, b, c\}\}$$

$$\mathcal{T}_P^{INT} \uparrow^1 = \mathcal{T}_P^{INT}(\mathcal{T}_P^{INT} \uparrow^0) = \{\{b\}, \{a, c\}, \{a, b\}, \{b, c\}, \{a, b, c\}\}$$

$$\mathcal{T}_P^{INT} \uparrow^\omega = \mathcal{T}_P^{INT} \uparrow^2 = \mathcal{T}_P^{INT}(\mathcal{T}_P^{INT} \uparrow^1) = \mathcal{T}_P^{INT} \uparrow^1$$

From which we obtain $\mathcal{T}_P^{coin} = \mathcal{T}_P^{INT} \uparrow^\omega = \{\{b\}, \{a, c\}, \{a, b\}, \{b, c\}, \{a, b, c\}\}$ and $\mathcal{TU}_P^{coin} = \{\{b\}, \{a, c\}, \{a\}, \{c\}, \{\}\}$. As expected according to Theorem 4, $\mathcal{T}_P^{coin} \cap \mathcal{TU}_P^{coin} = \mathcal{MM}_P(P)$.

In order to provide semantics for disjunctive logic programs we will resort to a Gelfond-Lifschitz like division operator [4], which transforms disjunctive logic programs into *PDLP*s. Then, to these programs, we can apply $\mathcal{T}_P^{coin}$ or $\mathcal{TU}_P^{coin}$ to determine, respectively, what is true and what is false by default.

Definition 11 (Division program). *Consider a disjunctive logic program P and an interpretation I. The division of program $\frac{P}{I}$ is the PDLP obtained by following the procedure below:*

- *For each rule $a_1 \vee \ldots \vee a_l \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n \in P$ such that $\{c_1, \ldots, c_n\} \cap I = \emptyset$, add $\frac{P}{I}$ the rule $a_1 \vee \ldots \vee a_l \leftarrow b_1, \ldots, b_m$.*
- *No other rule is in $\frac{P}{I}$.*

In a normal logic program P, the program division is employed to define the operators Γ_P^t and Γ_P^{tu} to obtain, respectively, what is true and what is not false by default in P, i.e. true or undefined. In the sequel, we enroll this notion and apply it to coins. However, in order to guarantee that $2_{exp}^{\mathcal{H}_{\mathcal{I}_P}}$ and $2_{imp}^{\mathcal{H}_{\mathcal{I}_P}}$ are, respectively, closed under Γ_P^t and under Γ_P^{tu}, we impose $\mathcal{H}_{\mathcal{I}_{\frac{P}{I}}}$ equal to $\mathcal{H}_{\mathcal{I}_P}$ by assuming P and $\frac{P}{I}$ defined over the same language.

Definition 12 (Γ_P^t and Γ_P^{tu}). *Let P be a disjunctive logic program and $\mathcal{I}^p =< \mathcal{I}^t, \mathcal{I}^{tu} >$ a partial coin. We define $\Gamma_P^t : 2_{imp}^{\mathcal{H}_{\mathcal{I}_P}} \rightarrow 2_{exp}^{\mathcal{H}_{\mathcal{I}_P}}$ and $\Gamma_P^{tu} : 2_{exp}^{\mathcal{H}_{\mathcal{I}_P}} \rightarrow 2_{imp}^{\mathcal{H}_{\mathcal{I}_P}}$ thus:*

$$\Gamma_P^t(\mathcal{I}^{tu}) = \bigcup_{I \in \mathcal{I}^{tu}} \mathcal{T}_{\frac{P}{I}}^{coin} \text{ and } \Gamma_P^{tu}(\mathcal{I}^t) = \bigcup_{I \in \mathcal{I}^t} \mathcal{TU}_{\frac{P}{I}}^{coin}$$

Casting attention to the definition of well founded semantics for normal logic programs [8], we now introduce the operator Φ_P to establish what follows immediately from a disjunctive logic program given a partial coin:

Definition 13 (Φ_P operator). *Consider a disjunctive logic program P and a partial coin $\mathcal{I}^p =< \mathcal{I}^t, \mathcal{I}^{tu} >$, where*

$$\mathfrak{F}(\mathcal{I}^{tu}) = \{A \in H_{B_P} \mid \forall I \in \mathcal{I}^{tu}, A \notin I\} \text{ and } \mathfrak{T}(\mathcal{I}^p) = \{I \in \mathcal{I}^t \mid \mathfrak{F}(\mathcal{I}^{tu}) \cap I = \emptyset\}.$$

We define

$$\Phi_P(\mathcal{I}^p) =< \Gamma^t_P(\mathcal{I}^{tu}), \Gamma^{tu}_P(\mathfrak{T}(\mathcal{I}^p)) >$$

Conceiving the partial coin $\mathcal{I}^p =< \mathcal{I}^t, \mathcal{I}^{tu} >$ as representing the safe knowledge so far, the rationale behind the Φ_P operator is that from $\mathcal{I}^{tu}$ one can conclude what definitely does not hold; so we can use this set of interpretations for deriving what surely holds. On the other hand, from $\mathcal{I}^t$ one knows what is true; then, by ignoring those interpretations containing atoms false by default in $\mathcal{I}^t$, one can use the remaining interpretations for deriving what may possibly hold. The Φ_P operator is assured monotonic with respect to $\sqsubseteq_k$:

Theorem 5 (Monotonicity of operator Φ_P according to $\sqsubseteq_k$). *Consider a disjunctive logic program P plus two partial coins $\mathcal{I}^p =< \mathcal{I}^t, \mathcal{I}^{tu} >$ and $\mathcal{J}^p =< \mathcal{J}^t, \mathcal{J}^{tu} >$. If $\mathcal{I}^p \sqsubseteq_k \mathcal{J}^p$ then $\Phi_P(\mathcal{I}^p) \sqsubseteq_k \Phi_P(\mathcal{J}^p)$.*

We ascertain forthwith, by the Knaster-Tarski theorem, that, for every disjunctive logic program P, Φ_P possesses a least fixed point under $\sqsubseteq_k$, thereby establishing the well-founded model with disjunction of P.

Definition 14 (Well-founded semantics with disjunction). *Let P be a disjunctive logic program. Each fixed point of the Φ_P operator corresponds to a partial stable model with disjunction of P. In particular, the least one under the $\sqsubseteq_k$ ordering is its well-founded model. It can be obtained by the following transfinite sequence of partial coins $\mathcal{I}^{\uparrow\alpha}_P =< \mathcal{I}^t_\alpha, \mathcal{I}^{tu}_\alpha >$:*

$$\begin{aligned}
&\mathcal{I}^{\uparrow 0}_P =< \mathcal{H}_{\mathcal{I}_P}, \mathcal{H}_{\mathcal{I}_P} > \\
&\mathcal{I}^{\uparrow\alpha}_P = \Phi_P(\mathcal{I}^{\uparrow\alpha-1}_P) =< \Gamma^t_P(\mathcal{I}^{tu}_{\alpha-1}), \Gamma^{tu}_P(\mathfrak{T}(\mathcal{I}^{\uparrow\alpha-1}_P)) >, \text{ for successor ordinal } \alpha \\
&\mathcal{I}^{\uparrow\alpha}_P =< lub_{exp}(\{\mathcal{I}^t_\beta \mid \beta < \alpha\}), glb_{imp}(\{\mathcal{I}^{tu}_\beta \mid \beta < \alpha\}) >= \\
&\quad < \textstyle\bigcap_{\beta<\alpha} \mathcal{I}^t_\beta, \bigcap_{\beta<\alpha} \mathcal{I}^{tu}_\beta >, \text{for limit ordinal } \alpha.
\end{aligned}$$

The Well-Founded Semantics with Disjunction for a disjunctive logic program P ($WFS_d(P)$) is defined by $\mathcal{I}^{\uparrow\alpha}_P$, where α is the smallest ordinal such that $\mathcal{I}^{\uparrow\alpha}_P = \Phi_P(\mathcal{I}^{\uparrow\alpha}_P)$.

It is noteworthy stressing that, for normal logic programs, WFS_d and WFS are isomorphic:

Theorem 6. *Let P be a normal logic program, $\mathcal{W} =< \mathcal{W}^t, \mathcal{W}^{tu} >$ be the WFS_d of P, and $W =< W^t, W^{tu} >$ be the WFS of P.*

- *An atom A is true in $\mathcal{W}$ iff it is also true in W.*
- *not A is true in $\mathcal{W}$ iff it is also true in W.*

Hence, for normal logic programs, WFS_d may be understood as an alternative characterisation of WFS, satisfying one condition we exacted for evaluating a good well-founded semantics for disjunctive logic programs.

One may wonder why not define Φ_P just as $< \Gamma_P^t(\mathcal{I}^{tu}), \Gamma_P^{tu}(\mathcal{I}^t)) >$. The justification is simple; posed in that manner, WFS_d becomes ineligible for one of the program transformations presented in [2]: *positive reduction*. We illustrate this in the next example:

Example 5. Let $P = \{\, a \vee b,\, b \leftarrow not\ c\}$

$\mathcal{I}_P^{\uparrow 0} = < \mathcal{H}_{\mathcal{I}_P}, \mathcal{H}_{\mathcal{I}_P} >$
$\mathcal{I}_P^{\uparrow 1} = < \{\{a\}, \{b\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}, \{\{a\}, \{b\}, \{\}\} >$
$\mathcal{I}_P^{\uparrow 2} = < \{\{b\}, \{a,b\}, \{b,c\}, \{a,b,c\}\}, \{\{b\}, \{\}\} >$
$WFS_d(P) = < \mathcal{W}^t, \mathcal{W}^{tu} > = \mathcal{I}_P^{\uparrow 3} = \mathcal{I}_P^{\uparrow 2}$

As *not c* is *true* in $WFS_d(P)$, $c \in \mathfrak{F}(\mathcal{W}^{tu})$, only interpretations in $\mathcal{W}^t$ deprived of c are considered to determine $\mathcal{W}^{tu}$, i.e.

$$\mathcal{W}^{tu} = \bigcup_{W_1 \in \mathfrak{T}(WFS_d(P))} \mathcal{TU}^{coin}_{\frac{P}{W_1}} \tag{5}$$

Considering that for every $W_1 \in \mathfrak{T}(WFS_d(P))$, $\frac{P}{W_1} = \{a \vee b, b\}$, we have $imp(min(\mathcal{T}^{coin}_{\frac{P}{W_1}})) = \{\{b\}, \{\}\}$, leading us to conclude *not a* is in $WFS_d(P)$.

Indeed, had we considered every interpretation in $\mathcal{W}^t$ to characterise $\mathcal{W}^{tu}$ in (5), then whether from $\{b,c\}$ or $\{a,b,c\}$ we would obtain $a \in \mathcal{TU}^{coin}_{\frac{P}{\{b,c\}}}$ or $a \in \mathcal{TU}^{coin}_{\frac{P}{\{a,b,c\}}}$, because the rule $b \leftarrow not\ c \in P$ is eliminated in $\frac{P}{\{b,c\}}$ as well as in $\frac{P}{\{a,b,c\}}$. Under such a scheme, we would infer the unexpected result $\{a\} \in \mathcal{W}^{tu}$, dismissing any possibility of concluding *not a* in $WFS_d(P)$.

4 Examples

By exhibiting appropriate examples, we are about to show that WFS_d does not coincide with the most prominent semantics for disjunctive logic programs [10, 13, 1, 11, 9, 2, 19].

Example 6. Let $P_1 = \{a \vee b \vee c, a \leftarrow not\ b, b \leftarrow not\ c, c \leftarrow not\ a\}$.

$\mathcal{I}_{P_1}^{\uparrow 0} = < \mathcal{H}_{\mathcal{I}_{P_1}}, \mathcal{H}_{\mathcal{I}_{P_1}} >$
$\mathcal{I}_{P_1}^{\uparrow 1} = < \{\{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}, \mathcal{H}_{\mathcal{I}_P} >$
$\mathcal{I}_{P_1}^{\uparrow 2} = < \{\{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\},$
$\quad \{\{a,b\}, \{b,c\}, \{a,c\}, \{a\}, \{b\}, \{c\}, \{\}\} >$
$WFS_d(P_1) = \mathcal{I}_{P_1}^{\uparrow 3} = \mathcal{I}_{P_1}^{\uparrow 2}.$

Program P_1 is not assigned any meaning according to Partial Disjunctive Stable Models [11]. In contradistinction, WFS_d enables us to say that both $a \vee b \vee c$ and $not\ a \vee not\ b \vee not\ c$ are true.

Indeed, even for stratified programs, where Partial Disjunctive Stable Models and Perfect Models [10] are always defined, we obtain distinct results via WFS_d:

Example 7. Let $P_2 = \{a \vee b, b \vee c, d \leftarrow not\ a, d \vee e \leftarrow not\ c\}$.

$$\begin{aligned}
&\mathcal{I}_{P_2}^{\uparrow 0} = <\mathcal{H}_{\mathcal{I}_{P_2}}, \mathcal{H}_{\mathcal{I}_{P_2}}> \\
&\mathcal{I}_{P_2}^{\uparrow 1} = <\{\{b\},\{a,c\},\{a,b\},\{b,c\},\{b,d\},\{b,e\},\{a,b,c\},\{a,b,d\},\{a,b,e\},\{a,c,d\},\\
&\qquad \{a,c,e\},\{b,c,d\},\{b,c,e\},\{b,d,e\},\{a,b,c,d\},\{a,b,c,e\},\{a,b,d,e\},\{b,c,d,e\},\\
&\qquad \{a,c,d,e\},\{a,b,c,d,e\}\},\\
&\qquad \{\{a,c,d\},\{a,c,e\},\{a,c\},\{a,d\},\{a,e\},\{c,d\},\{c,e\},\{b,d\},\{b,e\},\{a\},\{b\},\\
&\qquad \{c\},\{d\},\{e\},\{\}\}> \\
&WFS_d(P_2) = \mathcal{I}_{P_2}^{\uparrow 2} = \mathcal{I}_{P_2}^{\uparrow 1}.
\end{aligned}$$

P_2 has the partial stable models (perfect models): $<\{a,c\},\{a,c\}>$ and $<\{b,d\},\{b,d\}>$. Hence *not* e is obtained according to disjunctive partial stable models and perfect models; that same conclusion is verified under static [9] and stationary semantics, whilst it stands *undefined* in WFS_d.

In the next example we compare WFS_d to *GDWFS* [1], *D-WFS* [2], *WFDS* [19], and *SWFS* [13]:

Example 8. Let $P_3 = \{a \vee b \leftarrow not\ b, b \leftarrow not\ b\}$.

$$\begin{aligned}
&\mathcal{I}_{P_3}^{\uparrow 0} = <\mathcal{H}_{\mathcal{I}_{P_3}}, \mathcal{H}_{\mathcal{I}_{P_3}}> \\
&\mathcal{I}_{P_3}^{\uparrow 1} = <\{\{\},\{a\},\{b\},\{a,b\},\{a,b\}\},\{\{b\},\{\}\}> \\
&WFS_d(P_3) = \mathcal{I}_{P_3}^{\uparrow 2} = \mathcal{I}_{P_3}^{\uparrow 1}.
\end{aligned}$$

In P_3, WFS_d assigns *false* to a, whilst b remains *undefined.* Given that *D-WFS* deletes the first clause by elimination of non-minimal clauses, a is obtained as *false* and b continues *undefined* because there is no rule left with a in the head. *SWFS* assigns *undefined* to both a and b. In *GDWFS* b is *true* and a is *false.* Finally, for *WFDS* a and b are undefined. Then we have that in P_3 *GDWFS* derives the most information, followed by WFS_d and *D-WFS*. The weakest semantics for this program are *SWFS* and *WFDS*.

So far WFS_d and *D-WFS* have presented the same results. However this is not the case generally:

Example 9. Let $P_4 = \{a \vee b, c \leftarrow not\ a, c \leftarrow not\ b\}$.

$$\begin{aligned}
&\mathcal{I}_{P_4}^{\uparrow 0} = <\mathcal{H}_{\mathcal{I}_{P_4}}, \mathcal{H}_{\mathcal{I}_{P_4}}> \\
&\mathcal{I}_{P_4}^{\uparrow 1} = <\{\{a\},\{b\},\{a,b\},\{a,c\},\{b,c\},\{a,b,c\}\},\\
&\qquad \{\{a,c\},\{b,c\},\{a\},\{b\},\{c\},\{\}\}>
\end{aligned}$$

$\mathcal{I}_{P_4}^{\uparrow 2} = < \{\{a,c\},\{b,c\},\{a,b,c\}\},\{\{a,c\},\{b,c\},\{a\},\{b\},\{c\},\{\}\} >$
$WFS_d(P_4) = \mathcal{I}_{P_4}^{\uparrow 3} = \mathcal{I}_{P_4}^{\uparrow 2}$.

According to WFS_d, c is *true* in P_4 but *undefined* in $D\text{-}WFS$. Though $D\text{-}WFS$ and WFS_d do not coincide generally, yet $D\text{-}WFS$ is strictly weaker than WFS_d.

Theorem 7. *Let P_1 and P_2 be disjunctive logic programs such that P_2 results from P_1 by unfolding, elimination of tautologies and nonminimal rules, and positive and negative reduction. We have $WFS_d(P_1) = WFS_d(P_2)$.*

This guarantees that $D\text{-}WFS$ is strictly weaker than WFS_d, since $D\text{-}WFS$ is known to be the weakest semantics allowing unfolding, elimination of tautologies and nonminimal rules, and positive and negative reduction.

5 Conclusions and Future Work

When studying disjunctive logic programs, one of the first obstacles is not to be at a loss in keeping track of the alphabet soup of their different semantics. Despite the jocular tone of this affirmation, it reveals the great attention the logic programming community has devoted to this matter since 1982. Collaterally, it also reveals the difficulty in reaching a consensual semantics.

Bearing in mind that scores of semantics have already been presented to treat disjunction in logic programming, it might appear unreasonable to consider new proposals. Indeed, finding a semantics to capture the meaning of disjunctive logic programs is a conundrum sufficiently hard to dissuade simple solutions.

Notwithstanding, we have developed a new semantics for disjunctive logic programs, christened *Well-Founded Semantics with Disjunction* (WFS_d), by generalising to a set of interpretations the fixed point operator Ω_P defined in [8] to characterise *WFS*. Our semantics does not coincide generally with any other we have studied. It surmises $D\text{-}WFS$, Static, *GDWFS*, *WFDS* and *SWFS*. Yet we have shown WFS_d is uniquely defined for every disjunctive logic program, generalises *WFS*, and is strictly stronger than $D\text{-}WFS$.

Furthermore, preliminary results authorise us to state that WFS_d can be extended to deal with explicit negation, either explosive or paraconsistent. Again we postpone analyses of the algorithmic complexity of WFS_d, along with its model-theoretical counterpart, apropos of which we can promote a detailed scenario exploring the logical properties of WFS_d. In particular, it empowers us to identify the meaning assigned to disjunction and the properties supporting the relation between disjunction and default negation in our semantics.

Concomitantly we are working in another well-founded semantics for disjunctive logic programs we will present in a future work. Although preserving good properties of WFS_d, this new semantics is simpler and in some cases more credulous. For instance, in the program shown in Example 7, it concludes that *not e* is *true*, instead of standing *undefined* as in WFS_d.

Escorted by the aforementioned reasons, while we do not claim WFS_d is *the* best semantics for disjunctive logic programs, we steadfastly uphold that

WFS_d is not just a new semantics for disjunctive logic programs, but that it is at least a worthwhile one: a semantics whose adequacy rend it a possible choice for representing knowledge and reasoning in logic programming.

References

1. C. Baral, J. Lobo, and J. Minker. Generalized well-founded semantics for logic programs. In *10th International Conference on Automated Deduction*, 1990.
2. S. Brass and J. Dix. Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, 38(3):167–213, 1999.
3. J. A. Fernández and J. Minker. Bottom-up computation of perfect models for disjunctive theories. *Journal of Logic Programming*, 25(1):33–51, 1995.
4. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. A. Bowen, editors, *5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
5. J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.
6. J. Minker. On indefinite databases and the closed world assumption. In *6th Conference on Automated Deduction*, volume 138 of *Lecture Notes in Computer Science*, pages 292–308. Springer-Verlag, 1982.
7. J. Minker and A. Rajasekar. A fixpoint semantics for disjunctive logic programs. *Journal of Logic Programming*, 9(1):45–74, 1990.
8. H. Przymusinska and T. Przymusinski. Semantic issues in deductive databases and logic. In R. Banerji, editor, *Formal Techniques in Artificial Intelligence - a Sourcebook*, pages 321–367. North Holland, 1990.
9. T. Przymusinski. Static semantics for normal and disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, Special Issue on Disjunctive Programs:323–357, 1995.
10. T. C. Przymusinski. On the declarative semantics of stratified deductive databases and logic programs. In J. Minker, editor, *Foundation of Deductive Databases and Logic Programming*, pages 193–216. Morgan Kaufmann, 1988.
11. T. C. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing Journal*, 9:401–424, 1991.
12. R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 55–76. Plenum Press, 1978.
13. K. Ross. The well founded semantics for disjunctive logic programs. In *First International Conference on Deductive and Object Oriented Databases*, pages 352–369, 1989.
14. D. Seipel, J. Minker, and C. Ruiz. Model generation and state generation for disjunctive logic programs. *Journal of Logic Programming*, 32(1):49–69, 1997.
15. M. Smyth. Power domains. *Journal of Computer and System Sciences*, 16(1):23–36, 1978.
16. A. Tarski. Lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics*, pages 285–309, 1955.
17. M. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of ACM*, 4(23):733–742, 1976.
18. A. van Gelder, K. Ross, and J. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.
19. K. Wang. Argumentation-based abduction in disjunctive logic programming. *Journal of Logic Programming*, 2001.