TPLP **21** (5): 629–645, 2021.

doi:10.1017/S1471068421000235 First published online 23 September 2021

A Logical Characterization of the Preferred Models of Logic Programs with Ordered Disjunction

ANGELOS CHARALAMBIDIS, PANOS RONDOGIANNIS and ANTONIS TROUMPOUKIS
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece
(*e-mails:* a.charalambidis@di.uoa.gr, prondo@di.uoa.gr, antru@di.uoa.gr)

submitted 10 August 2021; accepted 20 August 2021

Abstract

Logic programs with ordered disjunction (LPODs) extend classical logic programs with the capability of expressing alternatives with decreasing degrees of preference in the heads of program rules. Despite the fact that the operational meaning of ordered disjunction is clear, there exists an important open issue regarding its semantics. In particular, there does not exist a *purely model-theoretic* approach for determining the *most preferred models* of an LPOD. At present, the selection of the most preferred models is performed using a technique that is not based exclusively on the models of the program and in certain cases produces counterintuitive results. We provide a novel, model-theoretic semantics for LPODs, which uses an additional truth value in order to identify the most preferred models of a program. We demonstrate that the proposed approach overcomes the shortcomings of the traditional semantics of LPODs. Moreover, the new approach can be used to define the semantics of a natural class of logic programs that can have both ordered and classical disjunctions in the heads of clauses. This allows programs that can express not only strict levels of preferences but also alternatives that are equally preferred.

KEYWORDS: ordered disjunction, answer sets, logic of here-and-there, preferences

1 Introduction

Logic programs with ordered disjunction (LPODs) extend classical logic programs with the capability of expressing ordered alternatives in the heads of program rules. In particular, LPODs allow the head of a program rule to be a formula $C_1 \times \cdots \times C_n$, where "$\times$" is a propositional connective called "ordered disjunction" and the C_i's are literals. The intuitive explanation of $C_1 \times \cdots \times C_n$ is "*I prefer C_1; however, if C_1 is impossible, I can accept C_2; $\cdots$; if all of $C_1, \ldots, C_{n-1}$ are impossible, I can accept C_n*". Due to their simplicity and expressiveness, LPODs are a widely accepted formalism for preferential reasoning, both in logic programming and in artificial intelligence.

At present, the semantics of LPODs is defined (Brewka 2002; Brewka *et al*. 2004) based on the answer set semantics, using a two-phase procedure. In the first phase, the answer sets of the LPOD are produced. This requires a modification of the standard definition of answer sets. In the second phase, the answer sets are "filtered", and we obtain the set of "*most preferred*" answer sets, which are taken as the meaning of the initial program. Notice that both phases are not purely model-theoretic: the first one requires the construction of the reduct of the program and the second one is performed using the so-called "*degree of satisfaction of rules*", a concept that relies on examining the rules of the program to justify the selection of the most preferred answer sets. Apart from its logical status, the current semantics of LPODs produces in certain cases counterintuitive most preferred answer sets. This discussion leads naturally to the question: "*Is it possible to specify the semantics of LPODs in a purely model-theoretic way?*".

An important first step in this direction was performed by Cabalar (2011), who used Equilibrium Logic (Pearce 1996) to logically characterize the answer sets produced in the first phase described above. However, to our knowledge, the second phase (namely the selection of the most preferred answer sets), has never been justified model-theoretically. We consider this as an important shortcoming in the theory of LPODs. Apart from its theoretical interest, this question also carries practical significance, because, as we are going to see, the present formalization of the second phase produces in certain cases counterintuitive (and in our opinion undesirable) results. The main contribution of the present paper is to *provide a purely model-theoretic characterization of the semantics of LPODs*. The more specific contributions are the following:

- We propose a new semantics for LPODs which uses an additional truth value in order to select as most preferred models those in which a top preference fails only if it is *impossible* to be satisfied. We demonstrate that the proposed approach overcomes the shortcomings of the traditional semantics of LPODs. In this way, the most preferred models of an LPOD can be characterized by a preferential ordering of its models.
- We demonstrate that our approach can be seamlessly extended to programs that allow both ordered and classical disjunctions in the heads of clauses. In particular, we define a natural class of such programs and demonstrate that all our results about LPODs transfer, with minimal modifications, to this new class. In this way we provide a clean semantics for a class of programs that can express not only strict levels of preference but also alternatives that are equally preferred.

Section 2 introduces LPODs and gives relevant background. Sections 3 and 4 describe the shortcomings of LPOD semantics and give an intuitive presentation of the proposed approach for overcoming these issues. The remaining sections give a technical exposition of our results. The proofs of all results have been moved to corresponding appendices.

2 Background on LPODs

LPODs are an extension of the logic programs introduced by Gelfond and Lifschitz (1991), called *extended logic programs*, which support two types of negation: default (denoted by *not*) and strong (denoted by $\neg$). Strong negation is useful in applications but it is not very essential from a semantics point of view: a literal $\neg A$ is semantically treated as an atom A'. For the basic notions regarding extended logic programs, we assume some familiarity with the work of Gelfond and Lifschitz (1991).

Definition 1
A (propositional) LPOD is a set of rules of the form:

$$C_1 \times \cdots \times C_n \leftarrow A_1, \ldots, A_m, \mathit{not}\, B_1, \ldots, \mathit{not}\, B_k,$$

where the C_i, A_j, and B_l are ground literals.

The intuitive explanation of a formula $C_1 \times \cdots \times C_n$ is "*I prefer C_1; however, if C_1 is impossible, I can accept C_2; $\cdots$; if all of $C_1, \dots, C_{n-1}$ are impossible, I can accept C_n*".

An interpretation of an LPOD is a set of literals. An interpretation is called *consistent* if there does not exist any atom A such that both A and $\neg A$ belong to I. The notion of model of an LPOD is defined as follows.

Definition 2
Let P be an LPOD and M an interpretation. Then, M is a model of P iff for every rule

$$C_1 \times \cdots \times C_n \leftarrow A_1, \dots, A_m, \mathit{not}\, B_1, \dots, \mathit{not}\, B_k,$$

if $\{A_1, \dots, A_m\} \subseteq M$ and $\{B_1, \dots, B_k\} \cap M = \emptyset$ then there exists $C_i \in M$.

To obtain the preferred answer sets of an LPOD, a two-phase procedure was introduced by Brewka (2002). In the first phase, the answer sets of the LPOD are produced. This requires a modification of the standard definition of answer sets for extended logic programs. In the second phase, the answer sets are "filtered", and we obtain the set of "most preferred" ones. The first phase is formally defined as follows.

Definition 3
Let P be an LPOD. The $\times$-reduct of a rule R of P of the form:

$$C_1 \times \cdots \times C_n \leftarrow A_1, \dots, A_m, \mathit{not}\, B_1, \dots, \mathit{not}\, B_k,$$

with respect to a set of literals I, is denoted by $R^I_\times$ and is defined as follows:

$$R^I_\times = \{C_i \leftarrow A_1, \dots, A_m \mid C_i \in I \text{ and } I \cap \{C_1, \dots, C_{i-1}, B_1, \dots, B_k\} = \emptyset\}.$$

The $\times$-reduct of P with respect to I is denoted by $P^I_\times$ and is the union of the reducts $R^I_\times$ for all R in P.

Definition 4
A set M of literals is an answer set of an LPOD P if M is a consistent model of P and M is the least model of $P^M_\times$.

The second phase produces the "most preferred" answer sets using the notion of the *degree of satisfaction of a rule*. Formally:

Definition 5
Let M be an answer set of an LPOD P. Then, M satisfies the rule:

$$C_1 \times \cdots \times C_n \leftarrow A_1, \dots, A_m, \mathit{not}\, B_1, \dots, \mathit{not}\, B_k,$$

- to degree 1 if $A_j \notin M$, for some j, or $B_i \in M$, for some i,
- to degree l, $1 \leq l \leq n$, if all $A_j \in M$, no $B_i \in M$, and $l = \min\{r \mid C_r \in M\}$

The degree of a rule R in the answer set M is denoted by $\mathit{deg}_M(R)$.

The satisfaction degrees of rules are then used to define a preference relation on the answer sets of a program. Given a set of literals M, let $M^i(P) = \{R \in P \mid \mathit{deg}_M(R) = i\}$. The preference relation is defined as follows.

Definition 6
Let M_1, M_2 be answer sets of an LPOD P. Then, M_1 is *inclusion-preferred* to M_2 iff there exists $k \geq 1$ such that $M^k_2(P) \subset M^k_1(P)$, and for all $j < k$, $M^j_2(P) = M^j_1(P)$.

Example 1
Consider the program:

```
wine × beer.
```

The above program has the answer sets {`wine`} and {`beer`}. The most preferred one is {`wine`} because it satisfies the unique fact of the program with degree 1 (while the answer set {`beer`} satisfies the fact with degree 2). Consider now the program:

```
wine × beer.
¬wine.
```

This has the unique answer set {`beer`} (the set {`wine,¬wine`} is rejected due to its inconsistency). Therefore, {`beer`} is also the most preferred answer set.

Notice that Brewka (2002) originally defined only the preference relation of Definition 6. In the follow-up paper (Brewka *et al.* 2004) two more preference relations were introduced, namely the *cardinality* and the *Pareto*, in order to treat cases for which the inclusion preference did not return the expected results. All these relations do not rely exclusively on the models of the source program, and are therefore subject to similar criticism. For this reason, in this paper we focus attention on the inclusion preference relation.

3 Some issues with the semantics of LPODs

From a foundational point of view, the main issue with the semantics of LPODs is that, in its present form, it is not purely model-theoretic. Despite the simplicity and expressiveness of ordered disjunction, one cannot characterize the meaning of a program by just looking at its set of models. Recall that this principle is one of the cornerstones of logic programming since its inception: the meaning of positive logic programs is captured by their minimum Herbrand model (van Emden and Kowalski 1976) and the meaning of extended logic programs is captured by their equilibrium models (Pearce 1996). How can the most preferred models of LPODs be captured model-theoretically? The existing issues regarding the semantics of LPODs are illustrated by the following examples.

3.1 The logical status of LPODs

Consider the following two programs:

```
a × b.
```

and:

```
b × a.
```

According to Definition 2, both programs have exactly the same models, namely {`a`}, {`b`}, and {`a,b`}. Moreover, both have the same answer sets, namely {`a`} and {`b`}. However, there is no model-theoretic explanation (namely one based on just the sets of models of the programs) of why the most preferred model of the first program is {`a`} while the most preferred model of the second program is {`b`}. As a conclusion, in order for the semantics of LPODs to be properly specified, a model-based approach needs to be devised.

3.2 Inaccurate preferential ordering of answer sets

Apart from the fact that Definition 5 is not purely model-theoretic, in many cases it also gives inaccurate preferential orderings of answer sets. Such inaccurate orderings have already been reported in the literature (Balduccini and Mellarkod 2003). The following program illustrates one such simple case.

Example 2
Consider the following program[1] whose declarative reading is "*I prefer to buy a Mercedes than a BMW. In case a Mercedes is available, I prefer a gas model to a diesel one. A gas model (of Mercedes) is not available*".

$$\begin{array}{l}\texttt{mercedes} \times \texttt{bmw}.\\ \texttt{gas\_mercedes} \times \texttt{diesel\_mercedes} \leftarrow \texttt{mercedes}.\\ \neg\texttt{gas\_mercedes}.\end{array}$$

The program has two answers sets: $M_1 = \{\texttt{mercedes}, \texttt{diesel\_mercedes}, \neg\texttt{gas\_mercedes}\}$ and $M_2 = \{\texttt{bmw}, \neg\texttt{gas\_mercedes}\}$. M_1 satisfies the first rule with degree 1, the second rule with degree 2, and the third rule with degree 1. M_2 satisfies the first rule with degree 2, the second rule with degree 1 (because the body of the rule evaluates to false), and the third rule with degree 1. According to Definition 6, the two answer sets are incomparable. However, it seems reasonable that the most preferred model is M_1: the first rule, which is a fact, specifies unconditionally a preference; the preferences of the second rule seem to be secondary, because they depend on the choice that will be made in the first rule.

The problems in the above example appear to be related to Definition 5: it assigns degree 1 in two cases that are apparently different: a rule that has a false body gets the same degree of satisfaction as a rule with a true body in whose head the first choice is satisfied. These are two different cases which, however, it is not obvious how to handle if we follow the satisfaction degree approach of Definition 5.

3.3 Unsatisfiable better options

It has been remarked (Brewka *et al*. 2004, discussion in page 342) that the inclusion preference is sensitive to the existence of *unsatisfiable better options*. The following example, taken from the paper by Brewka *et al*. (2004), motivates this problem.

Example 3
Assume we want to book accommodation for a conference (in the post-COVID era). We prefer a 3-star hotel from a 2-star hotel. Moreover, we prefer to be in walking distance from the conference venue. This can be modeled by the program:

$$\begin{array}{l}\texttt{walking} \times \neg\texttt{walking}.\\ \texttt{3-stars} \times \texttt{2-stars}.\end{array}$$

Consider now the scenario where the only available 3-star hotel (say $hotel_1$), is not in walking distance. Moreover, assume that the only available 2-star hotel (say $hotel_2$), happens to be in walking distance. According to Definition 6, these two options are incomparable because $hotel_1$ satisfies the first rule to degree 2 and the second rule to degree 1, while $hotel_2$ satisfies the first rule to degree 1 and the second rule to degree 2.

[1] Our example is identical (up to variable renaming) to an example given by Balduccini and Mellarkod (2003). This work was brought to our attention by one of the reviewers of the present paper.

Assume now that the above program is revised after learning that there also exists a 4-star hotel, which however is not an option for us (due to restrictions imposed by our funding agencies). The new program is:

```
walking × ¬walking.
4-stars × 3-stars × 2-stars.
¬4-stars.
```

In the new program, $hotel_1$ satisfies the first rule to degree 2 and the second rule to degree 2, while $hotel_2$ satisfies the first rule to degree 1 and the second rule to degree 3. According to Definition 6, $hotel_2$ is our preferred option.

The above example illustrates that under the "degree of satisfaction of rules" semantics, a small (and seemingly innocent) change in the program, can cause a radical change in the final preferred model. This sensitivity to changes is another undesirable consequence that stems from the fact that the second phase of the semantics of LPODs is not purely model-theoretic.

4 An intuitive overview of the proposed approach

The main purpose of this paper is to define a model-theoretic semantics for LPODs. In other words, we would like to be able to choose the most preferred answer sets of a program using preferential reasoning on the answer sets themselves. Actually, such an approach should also be applicable directly on the models of the source program, without the need to first construct the answer sets of the program. We would expect that such an approach would also provide solutions to the shortcomings of the previous section.

But on what grounds can we compare the answer sets (or, even better, the models) of a program and decide that some of them satisfy in a better way our preferences than the others? This seems like an impossible task because the answer sets (or the models) do not contain any information related to the ordered disjunction preferences of the program.

It turns out that we can introduce preferential information inside the answer sets of a program by slightly tweaking the underlying logic. The answer sets of extended logic programs are two-valued, i.e., a literal is either T (true) or F (false). We argue that in order to properly define the semantics of LPODs, we need a third truth value, which we denote by F^*. The intuitive reading of F^* is "*impossible to make true*".

To understand the need for F^*, consider again the intuitive meaning of $C_1 \times C_2$: we prefer C_2 *only if it is impossible for us to get* C_1. Impossible here means that if we try to make C_1 true, then the interpretation will become inconsistent. Therefore, we seem to need two types of false, namely F and F^*: F means "*false by default*" while F^* means "*impossible to make true*". The following example demonstrates these issues.

Example 4
Consider the program:

```
wine × beer.
¬wine.
```

As we are going to see in the coming sections, the most preferred answer set according to our approach is $\{(\texttt{wine}, F^*), (\texttt{beer}, T), (\neg\texttt{wine}, T)\}$. Notice that `wine` receives the value F^* because if we tried to make `wine` equal to T, the interpretation would become inconsistent (because `¬wine` is T). Notice also that, as we are going to see, the interpretation $\{(\texttt{wine}, F), (\texttt{beer}, T), (\neg\texttt{wine}, T)\}$ is not a model of the program.

The above discussion suggests the following semantics for "×" in the proposed logic. Let $u, v \in \{F, F^*, T\}$. Then:

$$u \times v = \begin{cases} v, & \text{if } u = F^* \\ u, & \text{otherwise} \end{cases}$$

The intuition of the above definition is that we return the value v *only if it is impossible to satisfy* u; in all other cases, we return u.

We now get to the issue of how we can use the value F^* to distinguish the most preferred answer sets: we simply identify those answer sets that are minimal with respect to their sets of atoms that have the value F^*. As we have mentioned, the value F^* means "*impossible to make true*". By minimizing with respect to the F^* values, *we only keep those answer sets in which a top preference fails only if it is impossible to be satisfied.*

The above discussion gives a description of how we can select the "*most preferred (three-valued) answer sets*" of an LPOD. However, it is natural to wonder whether it is possible to also characterize the answer sets model-theoretically, completely circumventing the construction of the reduct. A similar question was considered by Cabalar (2011), who demonstrated that the (two-valued) answer sets of an LPOD coincide with the equilibrium models of the program. We adapt Cabalar's characterization to fit in our setting. More specifically, we extend our three-valued logic to a four-valued one by adding a new truth value T^*, whose intuitive meaning is "*not false but its truth cannot be established*". We then demonstrate that the three-valued answer sets of an LPOD P are those models of P that are minimal with respect to a simple ordering relation and do not contain any T^* values. In this way we get a two-step, purely model-theoretic characterization of the most-preferred models of LPODs: in the first step we use the T^* values as a yardstick to identify those models that correspond to answer sets, and in the second step we select the most preferred ones, by minimizing with respect to the F^* values.

Finally, we consider the problem of characterizing the semantics of logic programs that contain both disjunctions and ordered disjunctions in the heads of rules. This is especially useful in cases where some of our preferences are equally important. For example:

```
(wine ∨ beer) × (soda ∨ juice).
```

states that `wine` and `beer` are our top preferences (but we have no preference among them), and `soda` and `juice` are our secondary preferences. We consider the class of programs in which the heads of rules consist of ordered disjunctions where each ordered disjunct is an ordinary disjunction (as in the above program). We demonstrate that the theory of these programs is very similar to that of LPODs. All our results for LPODs transfer with minimal modifications to this extended class of programs. This suggests that this is a natural class of programs that possibly deserves further investigation both in theory and in practice.

5 Redefining the answer sets of LPODs

In this section we provide a new definition of the answer sets of LPODs. The new definition is based on a three-valued logic which allows us to discriminate the most preferred answer sets using a purely model-theoretic approach. In Section 6 we demonstrate that by extending the logic to a four-valued one, we can identify directly the most preferred models of a program (without first producing the answer sets).

Definition 7
Let Σ be a nonempty set of propositional literals. The set of well-formed formulas is inductively defined as follows:

- Every element of Σ is a well-formed formula,
- The 0-place connective F^* is a well-formed formula,
- If ϕ_1 and ϕ_2 are well-formed formulas, then $(\phi_1 \wedge \phi_2)$, $(\phi_1 \vee \phi_2)$, $(\mathit{not}\, \phi_1)$, $(\phi_1 \leftarrow \phi_2)$, and $(\phi_1 \times \phi_2)$, are well-formed formulas.

The meaning of formulas is defined over the set of truth values $\{F, F^*, T\}$ which are ordered as $F < F^* < T$. Given two truth values v_1, v_2, we write $v_1 \leq v_2$ iff either $v_1 < v_2$ or $v_1 = v_2$.

Definition 8
A (three-valued) interpretation I is a function from Σ to the set $\{F, F^*, T\}$. We can extend I to apply to formulas, as follows:

$$\begin{array}{lcl} I(F^*) & = & F^* \\ I(\mathit{not}\, \phi) & = & \begin{cases} T, & \text{if } I(\phi) \leq F^* \\ F, & \text{otherwise} \end{cases} \\ I(\phi \leftarrow \psi) & = & \begin{cases} T, & \text{if } I(\phi) \geq I(\psi) \\ F, & \text{otherwise} \end{cases} \\ I(\phi_1 \wedge \phi_2) & = & \min\{I(\phi_1), I(\phi_2)\} \\ I(\phi_1 \vee \phi_2) & = & \max\{I(\phi_1), I(\phi_2)\} \\ I(\phi_1 \times \phi_2) & = & \begin{cases} I(\phi_2), & \text{if } I(\phi_1) = F^* \\ I(\phi_1), & \text{otherwise.} \end{cases} \end{array}$$

It is straightforward to see that the meanings of "$\vee$", "$\wedge$", and "$\times$" are associative and therefore we can write $I(\phi_1 \vee \cdots \vee \phi_n)$, $I(\phi_1 \wedge \cdots \wedge \phi_n)$, and $I(\phi_1 \times \cdots \times \phi_n)$ unambiguously (without the need of extra parentheses). Moreover, given literals $C_1, \ldots, C_n$ we will often write $I(C_1, \ldots, C_n)$ instead of $I(C_1 \wedge \cdots \wedge C_n)$.

The ordering $<$ (respectively, $\leq$) on truth values extends in the standard way on interpretations: given interpretations I_1, I_2 we write $I_1 < I_2$ (respectively, $I_1 \leq I_2$), if for all literals $L \in \Sigma$, $I_1(L) < I_2(L)$ (respectively, $I_1(L) \leq I_2(L)$).

When we consider interpretations of an LPOD program, we assume that the underlying set Σ is the set of literals of the program. The following definition will be needed.

Definition 9
An interpretation I is a *model* of an LPOD P if every rule of P evaluates to T under I. An interpretation I of P is called *consistent* if there do not exist literals A and $\neg A$ in P such that $I(A) = I(\neg A) = T$.

We can now give the new definitions for *reduct* and *answer sets* for LPODs.

Definition 10
Let P be an LPOD. The $\times$-reduct of a rule R of P of the form:

$$C_1 \times \cdots \times C_n \leftarrow A_1, \ldots, A_m, \mathit{not}\, B_1, \ldots, \mathit{not}\, B_k,$$

with respect to an interpretation I, is denoted by $R^I_\times$ and is defined as follows:

- If $I(B_i) = T$ for some i, $1 \leq i \leq k$, then $R^I_\times$ is the empty set.
- If $I(B_i) \neq T$ for all i, $1 \leq i \leq k$, then $R^I_\times$ is the set that contains the rules:

$$\begin{array}{lcl} C_1 & \leftarrow & F^*, A_1, \ldots, A_m \\ & \cdots & \\ C_{r-1} & \leftarrow & F^*, A_1, \ldots, A_m \\ C_r & \leftarrow & A_1, \ldots, A_m \end{array},$$

where r is the least index such that $I(C_1) = \cdots = I(C_{r-1}) = F^*$ and either $r = n$ or $I(C_r) \neq F^*$.

The $\times$-reduct of P with respect to I is denoted by $P^I_\times$ and is the union of the reducts $R^I_\times$ for all R in P.

The major difference of the above definition from that of Definition 3, are the clauses of the form $C_i \leftarrow F^*, A_1, \ldots, A_m$. These clauses are included so as that the value F^* can be produced for C_i when $I(A_1) = \cdots = I(A_m) = T$. Notice that if these clauses did not exist, there would be no way for the value F^* to be produced by the reduct.

Definition 11
Let P be an LPOD and M an interpretation of P. We say that M is a (three-valued) *answer set* of P if M is consistent and it is the $\leq$-least model of $P^M_\times$.

Notice that the least model of $P^M_\times$ in the above definition, can be constructed using the following immediate consequence operator $T_{P^M_\times} : (\Sigma \to \{F, F^*, T\}) \to (\Sigma \to \{F, F^*, T\})$:

$$T_{P^M_\times}(I)(C) = \max\{I(B_1, \ldots, B_n) \mid (C \leftarrow B_1, \ldots, B_n) \in P^M_\times\}.$$

Notice that since the set $\{F, F^*, T\}$ is a complete lattice under the ordering $\leq$, it is easy to see that the set of interpretations is also a complete lattice under the ordering $\leq$. Moreover, the operator $T_{P^M_\times}$ is monotonic over the complete lattice of interpretations; this follows from the fact that the meanings of conjunction (namely, min) and that of disjunction (namely, max), are monotonic. Then, by Tarski's fixed-point theorem, $T_{P^M_\times}$ has a least fixed-point, which can be easily shown to be the $\leq$-least model of $P^M_\times$.

The following lemma guarantees that our definition is a generalization of the well-known one for extended logic programs (Gelfond and Lifschitz 1991).

Lemma 1
Let P be a consistent extended logic program. Then the three-valued answer sets of P coincide with the standard answer sets of P.

The following lemmas, which hold for extended logic programs, also extend to LPODs.

Lemma 2
Let P be an LPOD and let M be an answer set of P. Then, M is a model of P.

Lemma 3
Let M be a model of an LPOD P. Then, M is a model of $P^M_\times$.

The answer sets of extended logic programs are minimal with respect to the classical truth ordering $F < T$. As it turns out, the answer sets of LPODs are minimal but with respect to an extended ordering, which is defined below.

Definition 12
The ordering $\prec$ on truth values is defined as follows: $F \prec T$ and $F \prec F^*$. For all $u, v \in \{F, F^*, T\}$, we write $u \preceq v$ if either $u \prec v$ or $u = v$. Given interpretations I_1, I_2, we write $I_1 \prec I_2$ (respectively, $I_1 \preceq I_2$) if for all literals $L \in \Sigma^*$, $I_1(L) \prec I_2(L)$ (respectively, $I_1(L) \preceq I_2(L)$).

Lemma 4
Every (three-valued) answer set M of an LPOD P, is a $\preceq$-minimal model of P.

As in the case of the original semantics of LPODs, we now need to define a preference relation over the answer sets of a program. Intuitively, we prefer those answer sets that maximize the prospect of satisfying our top choices in ordered disjunctions. This can be achieved by minimizing with respect to F^* values. More formally, we define the following ordering:

Definition 13
Let P be an LPOD and let M_1, M_2 be answer sets of P. Let M_1^* and M_2^* be the sets of literals in M_1 and M_2 respectively that have the value F^*. We say that M_1 is preferred to M_2, written $M_1 \sqsubset M_2$, if $M_1^* \subset M_2^*$.

Definition 14
An answer set of an LPOD P is called *most preferred* if it is minimal among all the answer sets of P with respect to the $\sqsubset$ relation.

The intuition behind the definition of $\sqsubset$ is that we prefer those answer sets that minimize the need for F^* values. In other words, an answer set will be most preferred if all the literals that get the value F^*, do this because there is no other option: these literals *must be false* in order for the program to have a model. We now examine the examples of Section 3 under the new semantics introduced in this section.

Example 5
Consider again the two programs discussed in Subsection 3.1. Under the proposed approach, the first program has two answer sets, namely $\{(\mathtt{a}, T), (\mathtt{b}, F)\}$ and $\{(\mathtt{a}, F^*), (\mathtt{b}, T)\}$, and the most preferred one (i.e., the minimal with respect to $\sqsubset$) is $\{(\mathtt{a}, T), (\mathtt{b}, F)\}$. The second program also has two answer sets, namely $\{(\mathtt{a}, F), (\mathtt{b}, T)\}$ and $\{(\mathtt{a}, T), (\mathtt{b}, F^*)\}$, and the most preferred one is $\{(\mathtt{a}, F), (\mathtt{b}, T)\}$. Notice that now the two programs have different sets of models and different answer sets and therefore it is reasonable that they have different most preferred ones.

Example 6
Consider the "cars" program of Subsection 3.2. It is easy to see that it has two answer sets, namely:

$$\begin{array}{rcl} M_1 & = & \{(\mathtt{mercedes}, T), (\mathtt{bmw}, F), (\mathtt{gas\_mercedes}, F^*), \\ & & \ (\mathtt{diesel\_mercedes}, T), (\neg\mathtt{gas\_mercedes}, T)\} \\ M_2 & = & \{(\mathtt{mercedes}, F^*), (\mathtt{bmw}, T), (\mathtt{gas\_mercedes}, F^*), \\ & & \ (\mathtt{diesel\_mercedes}, F^*), (\neg\mathtt{gas\_mercedes}, T)\} \end{array} .$$

According to the $\sqsubset$ ordering, the most preferred answer set is M_1.

Example 7
Consider the "hotels" program from Subsection 3.3. Under the restriction that there does not exist any 3-star hotel in walking distance and also there does not exist any 2-star hotel outside walking distance, we get the two incomparable answer sets:

$$\begin{aligned} M_1 &= \{(\texttt{walking}, T), (\neg\texttt{walking}, F), (\texttt{3-stars}, F^*), (\texttt{2-stars}, T)\}, \\ M_2 &= \{(\texttt{walking}, F^*), (\neg\texttt{walking}, T), (\texttt{3-stars}, T), (\texttt{2-stars}, F)\}. \end{aligned}$$

Consider now the modified program given in Subsection 3.3 (which contains the unsatisfiable better option of a 4-star hotel). Under the same restrictions as above, we get the two answer sets:

$$\begin{aligned} M_1' &= \{(\texttt{walking}, T), (\neg\texttt{walking}, F), (\texttt{3-stars}, F^*), (\texttt{2-stars}, T), \\ &\quad (\texttt{4-stars}, F^*), (\neg\texttt{4-stars}, T)\} \\ M_2' &= \{(\texttt{walking}, F^*), (\neg\texttt{walking}, T), (\texttt{3-stars}, T), (\texttt{2-stars}, F), \\ &\quad (\texttt{4-stars}, F^*), (\neg\texttt{4-stars}, T)\}. \end{aligned}$$

Under the proposed approach, the above two answer sets are also incomparable, and the problem identified in Subsection 3.3 no longer exists.

We close this section by stating a result that establishes a relationship between the answer sets produced by our approach (Definition 4) and those ones produced by the original formulation (Brewka 2002; Brewka *et al.* 2004).

Definition 15
Let I be a three-valued interpretation of LPOD P. We define $collapse(I)$ to be the set of literals L in P such that $I(L) = T$.

Lemma 5
Let P be an LPOD and M be a three-valued answer set of P. Then, $collapse(M)$ is an answer set of P according to Definition 4.

Lemma 6
Let N be an answer set of P according to Definition 4. There exists a unique three-valued interpretation M such that $N = collapse(M)$ and M is a three-valued answer set of P.

In other words, there is a bijection between the answer sets produced by our approach and the original ones. Moreover, each three-valued answer set only differs from the corresponding two-valued one in that some literals of the former may have a F^* value instead of an F value. However, these F^* values play an important role because they allow us to distinguish the most preferred answer sets.

6 A new logical characterization of the answer sets of LPODs

In this section we demonstrate that the answer sets of LPODs can be characterized in a purely logical way, namely without even the use of the reduct. In particular, we demonstrate that the answer sets of a given program P coincide with a well-defined subclass of the minimal models of P in a four-valued logic. This logic is an extension of the three-valued one introduced in Section 5 and minimality is defined with respect to a four-valued relation $\preceq$ that extends the three-valued one of Definition 12. The new logic is based on four truth values, ordered as follows:

$$F < F^* < T^* < T.$$

The value T^* can be read as "*not false but its truth cannot be established*". The connections of this logic with Equilibrium Logic (Pearce 1996) are discussed in Section 8.

An interpretation is now a function from Σ to the set $\{F, F^*, T^*, T\}$. The semantics of formulas with respect to an interpretation I is defined identically as in Definition 8.

$$\begin{array}{rclcrcl}
\phi \vee \phi & \equiv & \phi & \quad & \phi \times \phi & \equiv & \phi \\
\phi_1 \vee (\phi_2 \vee \phi_3) & \equiv & (\phi_1 \vee \phi_2) \vee \phi_3 & & \phi_1 \times (\phi_2 \times \phi_3) & \equiv & (\phi_1 \times \phi_2) \times \phi_3 \\
\mathit{not}\,(\phi_1 \vee \phi_2) & \equiv & \mathit{not}\,(\phi_1) \wedge \mathit{not}\,(\phi_2) & & \phi_1 \times \phi_2 \times \phi_1 & \equiv & \phi_1 \times \phi_2 \\
\mathit{not}\,(\phi_1 \wedge \phi_2) & \equiv & \mathit{not}\,(\phi_1) \vee \mathit{not}\,(\phi_2) & & \phi_1 \times (\phi_2 \vee \phi_3) & \equiv & (\phi_1 \times \phi_2) \vee (\phi_1 \times \phi_3) \\
\phi_1 \vee (\phi_2 \wedge \phi_3) & \equiv & (\phi_1 \vee \phi_2) \wedge (\phi_1 \vee \phi_3) & & \phi_1 \times (\phi_2 \wedge \phi_3) & \equiv & (\phi_1 \times \phi_2) \wedge (\phi_1 \times \phi_3)
\end{array}$$

Fig. 1. Equivalences in the 4-valued logic.

The notions of interpretation, consistent interpretation, and model are defined as in Definition 9. Moreover, we extend the three-valued $\preceq$ relation of Definition 12, as follows:

Definition 16
The (four-valued) ordering $\prec$ is defined as follows: $F \prec F^*$, $F \prec T^*$, $F \prec T$, and $T^* \prec T$. Given two truth values v_1, v_2, we write $v_1 \preceq v_2$ if either $v_1 \prec v_2$ or $v_1 = v_2$. Given interpretations I_1, I_2 of a program P, we write $I_1 \prec I_2$ (respectively, $I_1 \preceq I_2$) if for all literals L in P, $I_1(L) \prec I_2(L)$ (respectively, $I_1(L) \preceq I_2(L)$).

The following special kind of interpretations plays an important role in our logical characterization.

Definition 17
An interpretation I of LPOD P is called *solid* if for all literals L in P, it is $I(L) \neq T^*$.

We can now state the logical characterization of the answer sets of an LPOD.

Theorem 1
Let P be an LPOD. Then, M is a three-valued answer set of P iff M is a consistent $\preceq$-minimal model of P and M is solid.

In conclusion, given an LPOD we can purely logically characterize its most preferred models by first taking its consistent $\preceq$-minimal models that are solid and then keeping the $\sqsubset$-minimal ones (see Definitions 13 and 14).

An extended study of the properties of the proposed four-valued logic is outside the scope of the present paper. Figure 1 lists some useful equivalences; the first column is for classical connectives, while the second column contains equivalences involving the "$\times$" operator. We note the interaction of $\times$ with $\vee$ because this will be the central theme of the next section. It is easy to see that:

$$(\phi_1 \vee \phi_2) \times \phi_3 \not\equiv (\phi_1 \times \phi_3) \vee (\phi_2 \times \phi_3).$$

We note that this equivalence does not hold, a fact which will be referenced in the next section.

7 Answer sets of disjunctive LPODs

We now extend the ideas of the previous sections to programs that also allow standard disjunctions in the heads of rules. The case of disjunctive LPODs (DLPODs), was initially considered by Kärger *et al.* (2008) and reexamined by Cabalar (2011).

The main idea of using disjunctions in the heads of LPOD rules is described (Kärger *et al.* 2008) as follows: "*we use ordered disjunction to express preferences and disjunction to express indifferences*".

Example 8 (*taken from the paper by Kärger et al. (2008)*)
The program:

$$\texttt{pub} \times (\texttt{cinema} \ \vee \ \texttt{tv}).$$

expresses the fact that our top choice is going to the pub; if this is not possible, then our secondary preference can be satisfied by either going to the cinema or watching tv.

Kärger *et al.* (2008) consider rules whose heads are arbitrary combinations of atoms and the operators $\times$ and $\vee$. A set of transformations is then used in order to bring the heads of rules into "*Ordered Disjunctive Normal Form (ODNF)*". More specifically, each head is transformed into a formula of the form $\mathcal{C}_1 \vee \cdots \vee \mathcal{C}_n$ where each $\mathcal{C}_i$ is an ordered disjunction of literals. The resulting normalized rules are then used to obtain the preferred answer sets of the original program.

Example 9
The program of Example 8 is transformed to:

$$(\texttt{pub} \times \texttt{cinema}) \vee (\texttt{pub} \ \times \ \texttt{tv}).$$

This program is then used to get the preferred answer sets of the original one.

However, as observed by Cabalar (2011), one of the transformations used by Kärger *et al.* (2008) to obtain the ODNF, cannot be logically justified: the formula $(\phi_1 \vee \phi_2) \times \phi_3$ is not logically equivalent to the formula $(\phi_1 \times \phi_3) \vee (\phi_2 \times \phi_3)$ in terms of the logic of Here-and-There. As discussed at the end of Section 6, these two formulas are also not equivalent under our four-valued logic.

As an alternative approach to the semantics of DLPODs, Cabalar (2011) proposes to use the logical characterization on rules with heads that are arbitrary combinations of disjunctions and ordered disjunctions. We could extend this approach to get a logical characterization of the most preferred models of arbitrary DLPODs: given such a program P, we could at first consider all the $\preceq$-minimal models of P that are solid, and then select the $\sqsubset$-minimal among them. Such an approach is certainly general. However, we believe that not every such program carries computational intuition. A good example of this is given in Cabalar (2011, Example 2), where a program with both disjunction and ordered disjunction is given, and whose computational meaning is far from clear.

In the following, we define a class of programs which, as we claim, have a clear computational interpretation and at the same time retain *all* properties that we have identified for LPODs. Intuitively, we allow the head of a program rule to be a formula $\mathcal{C}_1 \times \cdots \times \mathcal{C}_n$ where each $\mathcal{C}_i$ is an ordinary disjunction of literals. Notice that the program in Example 8 belongs to this class, while the program in Example 9, does not. We believe that the programs of this class have a clear preferential interpretation. Intuitively, the rule heads of the programs we consider, denote a hierarchy of preferences imposed by the $\times$ operator; in each level of this hierarchy, we may have literals that have equal preference (this is expressed by standard disjunction).

It is important to stress that if we allowed arbitrary combinations of disjunctions and ordered disjunctions, the preferential intuition would be lost. To see this, consider for example the formula $(\texttt{a} \times \texttt{b}) \vee (\texttt{c} \times \texttt{d})$. This gives us the information that $(\texttt{a} \times \texttt{b})$ is at the same level of preference as $(\texttt{c} \times \texttt{d})$, and that $\texttt{a}$ is more preferred than $\texttt{b}$ and $\texttt{c}$ is more preferred than $\texttt{d}$; however, for example, it gives us no information of whether $\texttt{a}$ is more preferred than $\texttt{c}$. On the other hand, a formula of the fragment we consider, such as $(\texttt{a} \vee \texttt{b}) \times (\texttt{c} \vee \texttt{d})$ gives us a total order of $\texttt{a}$, $\texttt{b}$, $\texttt{c}$, and $\texttt{d}$.

Definition 18
A (propositional) DLPOD is a set of rules of the form:

$$\mathcal{C}_1 \times \cdots \times \mathcal{C}_n \leftarrow A_1, \ldots, A_m, \mathit{not}\, B_1, \ldots, \mathit{not}\, B_k,$$

where the A_j and B_l are ground literals and each $\mathcal{C}_i$ is a disjunction of ground literals.

As it turns out, the answer sets of such programs can be defined in an almost identical way as those of LPODs.

Definition 19
Let P be a DLPOD. The $\times$-reduct of a rule R of P of the form:

$$\mathcal{C}_1 \times \cdots \times \mathcal{C}_n \leftarrow A_1, \ldots, A_m, \mathit{not}\, B_1, \ldots, \mathit{not}\, B_k,$$

with respect to an interpretation I, is denoted by $R^I_\times$ and is defined as follows:

- If $I(B_i) = T$ for some i, $1 \leq i \leq k$, then $R^I_\times$ is the empty set.
- If $I(B_i) \neq T$ for all i, $1 \leq i \leq k$, then $R^I_\times$ is the set that contains the rules:

$$\begin{array}{lcl} \mathcal{C}_1 & \leftarrow & F^*, A_1, \ldots, A_m \\ & \cdots & \\ \mathcal{C}_{r-1} & \leftarrow & F^*, A_1, \ldots, A_m \\ \mathcal{C}_r & \leftarrow & A_1, \ldots, A_m \end{array},$$

where r is the least index such that $I(\mathcal{C}_1) = \cdots = I(\mathcal{C}_{r-1}) = F^*$ and either $r = n$ or $I(\mathcal{C}_r) \neq F^*$.

The $\times$-reduct of P with respect to I is denoted by $P^I_\times$ and is the union of the reducts $R^I_\times$ for all R in P.

Definition 20
Let P be a DLPOD and M a (three-valued) interpretation of P. Then, M is an *answer set* of P if M is consistent and M is a minimal model of the disjunctive program $P^M_\times$.

As it turns out, all the results we have obtained in the previous section, hold for DLPODs. The proofs of these results are (surprisingly) almost identical (modulo some minor notational differences) to the proofs of the corresponding results for LPODs. For reasons of completeness, the corresponding proofs are given in the supplementary material corresponding to this paper at the TPLP archives. The extended results are stated below.

Lemma 7
Let P be a consistent disjunctive extended logic program. Then, the answer sets of P according to Definition 20, coincide with the standard answer sets of P.

Lemma 8
Let P be a DLPOD and let M be an answer set of P. Then, M is a model of P.

Lemma 9
Let M be a model of a DLPOD P. Then, M is a model of $P^M_\times$.

Lemma 10
Every answer set M of a DLPOD P, is a $\preceq$-minimal model of P.

Theorem 2
Let P be a DLPOD. Then, M is an answer set of P iff M is a consistent $\preceq$-minimal model of P and M is solid.

The similarity of the definitions and of the theoretical results of DLPODs to those of standard LPODs, makes us believe that this is indeed an interesting class of programs that deserves further attention.

8 Related and future work

The work on LPODs is closely related to "*Qualitative Choice Logic*" (QCL) (Brewka *et al.* 2004). QCL is an extension of propositional logic with the preferential connective "$\times$", which has the same intuitive meaning as in LPODs: $A \times B$ is read "*if possible A, but if A is impossible then at least B*". Essentially, QCL is the propositional logic underlying LPODs. It is worth noting that the semantics of QCL is based on the "degree of satisfaction" of formulas, which is connected to the idea of the degree of satisfaction of the rules of LPODs (Definition 5). Moreover, as remarked by one of the reviewers of the present paper, the DLPODs introduced in Section 7 are closely connected to the "basic choice formulas" of QCL (Brewka *et al.* 2004, Section 3.1, Definition 8). It would be interesting to investigate whether our four-valued logic can be used to provide an alternative semantics for QCL.

The work reported in this paper is closely connected to the work of Cabalar (2011), who first considered the problem of expressing logically the semantics of LPODs. The key difference between the two works is that ours provides a characterization of *both* phases of the production of the most preferred models of an LPOD, while Cabalar's work concentrates on the first one.

It is important to stress here that both our work as well as the work of Cabalar (2011), are influenced by the work of Pearce (1996) who first gave a logical characterization of the answer sets of extended logic programs, using *Equilibrium Logic*. This is a non-monotonic logic which is defined on top of the monotonic logic of *Here-and-There* (Heyting 1930), using a model preference approach. The technique we have proposed in this paper, when applied to a consistent extended logic program P, produces the standard answer sets of P; this is a direct consequence of Theorem 1 and Lemma 1. Therefore, for extended logic programs, the Equilibrium Logic gives the same outcome as our approach which is based on a four-valued logic and $\preceq$-minimal models that are *solid*. We believe that a further investigation of the connections of our approach with that of Equilibrium Logic is a worthwhile topic.

Our work is the first to provide a purely model-theoretic characterization of the semantics of LPODs. To our knowledge, the four-valued logic we have utilized does not appear to be a well-known variant/extension of Here-and-There. However, some seemingly related logics have been used in the literature of answer set extensions. The original definition of Equilibrium Logic included a second constructive negation, which corresponds to Nelson's strong negation (Nelson 1949). This gave rise to a five-valued extension of Here-and-There, called $\mathcal{N}_5$. Also, a logic called $\mathcal{X}_5$, that is closely connected to $\mathcal{N}_5$, was recently proposed by Aguado *et al.* (2019) in order to capture the semantics of arbitrary combinations of explicit negation with nested expressions. Both $\mathcal{N}_5$ and $\mathcal{X}_5$ appear to be connected to our four-valued logic due to the different notions of false and true that they employ in order to capture aspects that arise in answer set semantics. However, the ordering of the truth values and the semantics of the logical connectives are different, and the exact correspondence (if any) between these logics and the present one, is not straightforward to establish. This is certainly an interesting topic for further investigation.

Another promising topic for future work is the characterization of the notion of *strong equivalence* (Lifschitz *et al.* 2001) for LPODs and DLPODs. When two logic programs are strongly equivalent, we can replace one for the other inside a bigger program without worrying that the semantics of the bigger program will be affected. Characterizations of strong equivalence for LPODs have already been obtained by Faber *et al.* (2008). It would be interesting to investigate if the logical characterization of the semantics of LPODs and DLPODs developed in the present paper, can offer advantages compared with their work.

Acknowledgments

We would like to thank the three anonymous reviewers of our paper for their careful and insightful comments.

This research is co-financed by Greece and the European Union (European Social Fund- ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014- 2020" in the context of the project "Techniques for implementing qualitative preferences in deductive querying systems" (5048151).

Supplementary material

To view supplementary material for this article, please visit http://dx.doi.org/10.1017/S1471068421000235.

References

Aguado, F., Cabalar, P., Fandinno, J., Pearce, D., Pérez, G. and Vidal, C. 2019. Revisiting explicit negation in answer set programming. *Theory and Practice of Logic Programming 19,* 5–6, 908–924.

Balduccini, M. and Mellarkod, V. S. 2003. Cr-prolog with ordered disjunction. In *Answer Set Programming, Advances in Theory and Implementation, Proceedings of the 2nd Intl. ASP 2003 Workshop, Messina, Italy, September 26–28, 2003*. CEUR Workshop Proceedings, vol. 78. CEUR-WS.org.

Brewka, G. 2002. Logic programming with ordered disjunction. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 – August 1, 2002, Edmonton, Alberta, Canada.* AAAI Press/The MIT Press, 100–105.

Brewka, G., Benferhat, S. and Berre, D. L. 2004. Qualitative choice logic. *Artificial Intelligence 157,* 1–2, 203–237.

Brewka, G., Niemelä, I. and Syrjänen, T. 2004. Logic programs with ordered disjunction. *Computational Intelligence 20,* 2, 335–357.

Cabalar, P. 2011. A logical characterisation of ordered disjunction. *AI Communication 24,* 2, 165–175.

Faber, W., Tompits, H. and Woltran, S. 2008. Notions of strong equivalence for logic programs with ordered disjunction. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, G. Brewka and J. Lang, Eds. AAAI Press, 433–443.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9,* 3/4, 365–386.

Heyting, A. 1930. Die formalen regeln der intuitionistischen logik. *Sitzungsbericht PreuBische Akademie der Wissenschaften Berlin, physikalisch-mathematische Klasse II*, 42–56.

Kärger, P., Lopes, N., Olmedilla, D. and Polleres, A. 2008. Towards logic programs with ordered and unordered disjunction. In *Proceedings of Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP2008), 24th International Conference on Logic Programming (ICLP 2008)*. 46–60.

Lifschitz, V., Pearce, D. and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic 2,* 4, 526–541.

Nelson, D. 1949. Constructible falsity. *The Journal of Symbolic Logic 14,* 1, 16–26.

Pearce, D. 1996. A new logical characterisation of stable models and answer sets. In *Non-Monotonic Extensions of Logic Programming, NMELP 1996, Bad Honnef, Germany, September 5–6, 1996, Selected Papers*. Lecture Notes in Computer Science, vol. 1216. Springer, 57–70.

van Emden, M. H. and Kowalski, R. A. 1976. The semantics of predicate logic as a programming language. *Journal of ACM 23,* 4, 733–742.