TPLP **22** (5): 708–722, 2022.

doi:10.1017/S1471068422000242 First published online 1 July 2022

Strong Equivalence of Logic Programs with Ordered Disjunction: A Logical Perspective

ANGELOS CHARALAMBIDIS
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece
(*e-mail:* acharal@iit.demokritos.gr)

CHRISTOS NOMIKOS
Department of Computer Science and Engineering, University of Ioannina, Greece
(*e-mail:* a.charalambidis@di.uoa.gr)

PANOS RONDOGIANNIS
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece
(*e-mail:* prondo@di.uoa.gr)

submitted 11 May 2022; accepted 8 June 2022;

Abstract

Logic Programs with Ordered Disjunction (LPODs) extend classical logic programs with the capability of expressing preferential disjunctions in the heads of program rules. The initial semantics of LPODs, although simple and quite intuitive, is not purely model-theoretic. As a result, certain properties of programs appear non-trivial to formalize in purely logical terms. For example, the current characterization of strong equivalence for LPODs, does not coincide with logical equivalence in some specific logic. This comes in sharp contrast with the well-known characterization of strong equivalence for classical logic programs, which coincides with logical equivalence in the logic of here-and-there. In this paper we obtain a purely logical characterization of strong equivalence for LPODs as logical equivalence in a four-valued logic. Moreover, we provide a new proof of the coNP-completeness of strong equivalence for LPODs, which has an interest in its own right since it relies on the special structure of such programs. Our results are based on the recent logical semantics of LPODs, a fact which we believe indicates that this new semantics may prove to be a useful tool in the further study of LPODs.

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

1 Introduction

Logic Programs with Ordered Disjunction (LPODs) (Brewka 2002; Brewka *et al.* 2004b) extend classical logic programs with the capability of expressing preferential disjunctions in the heads of program rules. The head of an LPOD rule is a formula $C_1 \times \cdots \times C_n$ intuitively understood as follows: "*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". The meaning of LPODs is

expressed by their *most-preferred answer sets* (Brewka 2002; Brewka *et al.* 2004b), namely a subset of their answer sets which satisfies in the best possible way the preferences in the head of program rules. Due to their elegance and expressiveness, LPODs are widely accepted as a concise and powerful formalism for preferential reasoning, both in logic programming and in artificial intelligence.

Although simple and quite intuitive, the original semantics of LPODs (Brewka 2002; Brewka *et al.* 2004b) is not purely model-theoretic. More specifically, the most-preferred answer sets of a program cannot be determined by just examining the set of models of the program. Instead, one has to additionally use an ordering relation which relies on the syntax of the source program. There have been reported in the literature (Balduccini and Mellarkod 2003; Brewka *et al.* 2004b; Charalambidis *et al.* 2021) cases where the original semantics of LPODs produces counterintuitive results. Another consequence of this semantics, is that certain properties of LPODs appear nontrivial to formalize in purely logical terms. In this paper we identify one such case, namely the problem of characterizing the notion of *strong equivalence* for LPODs.

The concept of strong equivalence for logic programs was introduced by Lifschitz *et al.* (2001) and has proven to be an essential and extensively studied property in ASP. Two logic programs P_1 and P_2 are termed strongly equivalent under a given semantics if for every logic program P, $P_1 \cup P$ has the same meaning as $P_2 \cup P$ under this given semantics. Obviously, when two logic programs are strongly equivalent, we can replace one for the other inside a bigger program without any change in the observable behavior of this program. Lifschitz *et al.* (2001) demonstrated that two programs are strongly equivalent under the answer set semantics (Gelfond and Lifschitz 1988) if and only if they are equivalent in the logic of here-and-there (Pearce 1996; 1999). The importance of this result stems from the fact that it relates the observable behavior of programs with a purely logical notion, namely that of logical equivalence.

Due to the significance of strong equivalence, it appears as a natural endeavor to study this concept for various extensions of logic programs. Shortly after the inception of LPODs, an exhaustive study of various notions of strong equivalence for LPODs was undertaken by Faber *et al.* (2008). Although the results of Faber *et al.* (2008) are accurately developed, they fall short of characterizing strong equivalence of LPODs as logical equivalence in some specific logic. This comes in sharp contrast with the aforementioned characterization of strong equivalence for classical logic programs as logical equivalence in the logic of here-and-there. We believe that this is not an inherent shortcoming of the work of Faber *et al.* (2008), but instead a possibly unavoidable consequence of the fact that the original semantics of LPODs is not purely model-theoretic.

Recently, a purely model-theoretic semantics for LPODs was developed by Charalambidis *et al.* (2021), who undertook a question initially posed by Cabalar (2011). More specifically, as it is demonstrated by Charalambidis *et al.* (2021), the most-preferred answer sets of an LPOD can be obtained as the least models of the program under a novel four-valued logic, using an ordering relation that is independent of the syntax of the program. It is also demonstrated that the shortcomings of LPODs that have been observed in the literature (Balduccini and Mellarkod 2003; Brewka *et al.* 2004b; Charalambidis *et al.* 2021), are remedied by resorting to this new approach, and it is claimed that this new semantics may prove helpful in formalizing, in purely logical terms, properties, and transformations of LPODs. It is therefore natural to wonder if this new semantic

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

characterization leads to a purely logical definition of strong equivalence for LPODs. The present paper investigates exactly this question. More specifically, the main contributions of the present paper are as follows:

- Following the work of Faber *et al.* (2008), we consider two alternative definitions of strong equivalence for LPODs, which can be supported under the model-theoretic framework developed by Charalambidis *et al.* (2021). We demonstrate that both of them coincide with the notion of logical equivalence of programs in the four-valued logic of Charalambidis *et al.* (2021). Our characterization gracefully extends the results of Lifschitz *et al.* (2001) for normal logic programs.
- We provide a new proof of the **coNP**-completeness of strong equivalence for LPODs, which has an interest in its own right, since it relies on the special structure of such programs. More specifically, the proof demonstrates **coNP**-hardness by a direct (and quite simple) reduction from 3SAT, without resorting to the well-known (and more involved) **coNP**-hardness result of Lin (2002) for strong equivalence of normal logic programs.

The rest of the paper is organized as follows. Section 2 provides the mathematical preliminaries that will be needed throughout the paper. Section 3 presents the characterization results for strong equivalence of LPODs. In Section 4 the **coNP**-completeness of strong equivalence for LPODs is established. Section 5 discusses related work and gives pointers for future work. The proofs of certain results have been moved in the supplementary material corresponding to this paper at the TPLP archives.

2 Background

In this section we present the necessary background that will be used throughout the paper. We start by defining the syntax and the semantics of the four-valued logic introduced by Charalambidis *et al.* (2021) and discuss how this logic can be used to redefine the semantics of LPODs.

Similarly to the paper by Faber *et al.* (2008), we do not consider strong negation, for reasons of simplicity.

Definition 1
Let Σ be a nonempty, countably infinite, set of propositional atoms. The set of *well-formed formulas* is inductively defined as follows:

- Every element of Σ is a well-formed formula,
- If ϕ_1 and ϕ_2 are well-formed formulas, then $(\phi_1 \wedge \phi_2)$, $(\phi_1 \vee \phi_2)$, $(not\, \phi_1)$, $(\phi_1 \leftarrow \phi_2)$, and $(\phi_1 \times \phi_2)$ are well-formed formulas.

We will use capital variables, like A, B, C, D, and their subscripted versions, to denote atoms; we will use L, and its subscripted versions, to denote literals (namely, atoms or negated atoms).

In order to define the semantics of well-formed formulas, we use the set $V = \{F, F^*, T^*, T\}$ of truth values, which are ordered as follows:

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

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

Definition 2
An *interpretation* I is a function from Σ to V. We can extend I to apply to formulas, as follows:

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

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 $L_1, \ldots, L_n$, we will often write $L_1, \ldots, L_n$ instead of $L_1 \wedge \cdots \wedge L_n$.

LPODs are sets of formulas of a special kind, specified by the following definition.

Definition 3
An LPOD is a finite set of rules of the form:

$$C_1 \times \cdots \times C_n \leftarrow A_1, \ldots, A_m, not\ B_1, \ldots, not\ B_k.$$

where $n \geq 1$, $m, k \geq 0$, and the C_i, A_j, and B_l are atoms.

We will use capital letters like P, Q, and their subscripted versions, to denote LPODs.

Definition 4
An interpretation I is a *model* of an LPOD P if every rule of P evaluates to T under I. Two LPODs are termed *logically equivalent* if they have the same models.

Charalambidis *et al.* (2021) defined the semantics of LPODs, namely the precise characterization of their most-preferred answer sets, based on the above four-valued logic. More specifically, the most-preferred answer sets of an LPOD are generated using a two-step procedure. In the first step, a subset of the models of the program is selected using a minimization procedure according to an ordering relation $\preceq$ defined below. These models are called *answer sets* of the given LPOD, because they can also be produced using a reduct-based approach similar to the one defined in the paper by Brewka *et al.* (2004b). In the second step, a subset of the answer sets is selected using a minimization procedure that examines the set of atoms that have the value F^* in each answer set. These two steps are formally defined below.

Definition 5
The ordering $\prec$ on truth values 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 $I_1 \preceq I_2$ if for all atoms A in P, $I_1(A) \preceq I_2(A)$. We write $I_1 \prec I_2$ if $I_1 \preceq I_2$ but $I_1 \neq I_2$.

It is easy to verify that $\preceq$ is a partial order.

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

Definition 6
An interpretation I of LPOD P is called *solid* if for all atoms A in P, it is $I(A) \neq T^*$.

Definition 7
An interpretation M of an LPOD P will be called an *answer set* of P if M is a $\preceq$-minimal model of P and M is solid.

Definition 8
Let P be an LPOD and let M_1 and M_2 be answer sets of P. Let M_1^* and M_2^* be the sets of atoms 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 9
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.

Example 1 (*taken from the paper by Charalambidis et al. (2021)*)
Consider the following program 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*".

```
mercedes × bmw ←
gas_mercedes × diesel_mercedes ← mercedes
false ← gas_mercedes, not false
```

The last clause is a standard technique in ASP in order to state that an atom (`gas_mercedes` in our case) is not true. The above program has two answer sets, namely:

$\{(\texttt{mercedes}, T), (\texttt{bmw}, F), (\texttt{gas\_mercedes}, F^*), (\texttt{diesel\_mercedes}, T), (\texttt{false}, F^*)\}$
$\{(\texttt{mercedes}, F^*), (\texttt{bmw}, T), (\texttt{gas\_mercedes}, F^*), (\texttt{diesel\_mercedes}, F^*), (\texttt{false}, F^*)\}$

According to the $\sqsubset$ ordering, the most-preferred answer set is the first one because it minimizes the F^* values. It is worth noting that under the original semantics of LPODs (Brewka 2002; Brewka *et al.* 2004b) two answer sets are produced that are incomparable (and therefore they are both considered as "most-preferred").

3 A logical characterization of strong equivalence for LPODs

In this section we establish a new, purely logical characterization of strong equivalence for LPODs. Our investigation has as a starting point the work of Faber *et al.* (2008), in which an exhaustive study of different forms of strong equivalence for LPODs was performed. Not all forms of strong equivalence studied by Faber *et al.* (2008) are applicable in our case. An explanation of this state of affairs and a detailed comparison of our technique with that of Faber *et al.* (2008) is given in Section 5. In our work, we examine two notions of strong equivalence, namely *strong equivalence under the most-preferred answer sets*, and *strong equivalence under all the answer sets*[1]. We demonstrate that these notions

[1] These two notions roughly correspond to the relations $\equiv^i_{s,\times}$ and $\equiv_{s,\times}$ defined in the paper by Faber *et al.* (2008).

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

can be captured by establishing logical equivalence in the four-valued logic of Section 2 of the programs involved.

Definition 10
Two LPODs P_1 and P_2 are termed *strongly equivalent under the most-preferred answer sets* if for every LPOD P, $P_1 \cup P$ and $P_2 \cup P$ have the same most-preferred answer sets.

Theorem 1
Two LPODs P_1 and P_2 are strongly equivalent under the most-preferred answer sets if and only if they are logically equivalent in four-valued logic.

Proof
($\Leftarrow$) Assume that P_1 and P_2 are logically equivalent in four-valued logic. Then, every four-valued model that satisfies one of them, also satisfies the other. This means that for all programs P, $P_1 \cup P$ has the same models as $P_2 \cup P$. But then, $P_1 \cup P$ has the same most-preferred answer sets as $P_2 \cup P$ (because the most-preferred answer sets of a program depend only on the set of all the models of the program). Therefore, $P_1 \cup P$ and $P_2 \cup P$ are strongly equivalent under the most-preferred answer sets.
($\Rightarrow$) Assume that P_1 and P_2 are strongly equivalent. Suppose that P_1 has a model M which is not a model of P_2. Without loss of generality, we may assume that $M(A) = F$, for every atom A in Σ that does not occur in $P_1 \cup P_2$.

We will show that we can construct an interpretation M', and a program P such that M' is a most-preferred answer set of one of $P_1 \cup P$ and $P_2 \cup P$ but not of the other, contradicting our assumption of strong equivalence.

First, we construct two sets of atoms that will help us define P. In particular, we construct two sets of atoms $\mathcal{T}$ and $\mathcal{F}$ each one containing a new atom for every A in P such that $M(A) = F^*$. More formally, let $\mathcal{T} = \{t_A \mid M(A) = F^*\}$ and $\mathcal{F} = \{f_A \mid M(A) = F^*\}$, where all t_A and f_A do not appear in P_1 and P_2. We define M' as:

$$M'(A) = \begin{cases} T & M(A) = T^* \\ T & A \in \mathcal{T} \\ F^* & A \in \mathcal{F} \\ M(A) & \text{otherwise} \end{cases}.$$

We claim that M' is a model of P_1. To verify this, take any rule in P_1 of the form

$$C_1 \times \cdots \times C_n \leftarrow A_1, \ldots, A_m, not\, B_1, \ldots, not\, B_k.$$

If $M(A_1, \ldots, A_m, not\, B_1, \ldots, not\, B_k) \geq T^*$, then it is also $M(C_1 \times \cdots \times C_n) \geq T^*$, since M is a model of P_1. Then, there exists $j \leq n$ such that $M(C_i) = F^*$ for all $i < j$, and $M(C_j) \geq T^*$. It follows that $M'(C_i) = F^*$ for all $i < j$, and $M'(C_j) = T$, which implies $M'(C_1 \times \cdots \times C_n) = T$. Therefore, M' satisfies the rule in this case.

If $M(A_1, \ldots, A_m, not\, B_1, \ldots, not\, B_k) = F^*$, then there exists A_i such that $M(A_i) = F^*$. By the definition of M', $M'(A_i) = F^*$, and thus $M'(A_1, \ldots, A_m, not\, B_1, \ldots, not\, B_k) \leq F^*$. Since M is a model of P_1 it satisfies the given rule and thus $M(C_1 \times \cdots \times C_n) \geq F^*$. If $M(C_1 \times \cdots \times C_n) = F^*$, then for all C_i, $M(C_i) = F^*$, which implies that $M'(C_i) = F^*$ and therefore $M'(C_1 \times \cdots \times C_n) = F^*$. If $M(C_1 \times \cdots \times C_n) > F^*$, then there exists $j \leq n$ such that for all $i < j$, $M(C_i) = F^*$

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

and $M(C_j) > F^*$, which implies that for all $i < j$, $M'(C_i) = F^*$ and $M'(C_j) > F^*$, and therefore $M'(C_1 \times \cdots \times C_n) > F^*$. In both cases, M' satisfies the given rule.

If $M(A_1, \ldots, A_m, not\, B_1, \ldots, not\, B_k) = F$, then either there exists A_i such that $M(A_i) = F$ or there exists B_j such that $M(B_j) \geq T^*$. It follows, by the definition of M', that $M'(A_i) = F$ or $M'(B_j) = T$ and as a result $M'(A_1, \ldots, A_m, not\, B_1, \ldots, not\, B_k) = F$. Therefore M' satisfies the rule in this case. Thus, M' is a model of P_1.

We proceed by distinguishing two cases that depend on whether M' is a model of P_2 or not.

Case 1: M' is not a model of P_2. We take:

$$P = \{A \leftarrow| \; M'(A) = T\} \cup \{A \times t_A \leftarrow| \; t_A \in \mathcal{T}\} \cup \{f_A \leftarrow not\, f_A, A \mid f_A \in \mathcal{F}\}$$

We claim that every model N of $\{A \times t_A \leftarrow| \; t_A \in \mathcal{T}\} \cup \{f_A \leftarrow not\, f_A, A \mid f_A \in \mathcal{F}\}$ has the following property:

$$\text{for every atom } A, \text{ if } M'(A) = F^*, \text{ then } N(A) \neq F. \tag{P1}$$

In order to prove our claim we distinguish two cases for atoms such that $M'(A) = F^*$: the atoms where $M(A) = F^*$ and the atoms in $\mathcal{F}$. For the first case, assume that for some A it is $M(A) = F^*$ and $N(A) = F$. But then there exists a rule $A \times t_A \leftarrow$ in P which is not satisfied by N, which is a contradiction. So, for all such atoms A it should be $N(A) \geq F^*$. For the second case, assume that for some f_A, it is $N(f_A) = F$. Then, the rule $f_A \leftarrow not\, f_A, A$ is not satisfied by N (since $N(A) \geq F^*$), which is also a contradiction. Therefore, our claim holds.

Now, it is easy to see that M' is a model of P and therefore a model of $P_1 \cup P$. Moreover, it is a most-preferred answer set of $P_1 \cup P$. Indeed, let N be a model of $P_1 \cup P$ and $N \prec M'$. Since M' does not assign any T^*, there exists A such that either $M'(A) = T$ and $N(A) \prec T$ or $M'(A) = F^*$ and $N(A) = F$. In the first case, P contains a fact $A \leftarrow$, which is not satisfied by N. In the second case, N does not satisfy property P1. In both cases, N is not a model of P, which is a contradiction. It follows that M' is $\preceq$-minimal model of $P_1 \cup P$.

Assume now that there exists some N which is a most-preferred answer set of $P_1 \cup P$ and $N \sqsubset M'$. There must exist some atom A such that $M'(A) = F^*$ and $N(A) \neq F^*$. We will show in the following that it should be $N(A) = T$ for those atoms. First, notice that $N(A) \neq T^*$ because since N is most-preferred it is also solid. Also, it must be $N(A) \neq F$, since N is a model of P and thus it satisfies property P1. Therefore, $N(A) = T$ for the atoms such that $M'(A) = F^*$ and $N(A) \neq F^*$.

However, we now claim that N is not $\preceq$-minimal. Indeed, we can construct N' from N that is also a model of $P_1 \cup P$ and $N' \prec N$. Define N' as:

$$N'(A) = \begin{cases} T^* & \text{if } A \in \mathcal{F} \text{ and } N(A) = T \\ N(A) & \text{otherwise} \end{cases}.$$

First, we need to establish that $N' \prec N$, that is, there exists atom A such that $N'(A) \prec N(A)$. By the assumption $N \sqsubset M'$ we know that there exists atom A such that $N(A) \neq M'(A)$ and we have established that $M'(A) = F^*$ and $N(A) = T$. The first case is for the atom A to be $A = f_B \in \mathcal{F}$ for some B, and it is straightforward that $N(f_B) = T$ and $N'(f_B) = T^*$. The second case is for A to be an atom such that $M(A) = F^*$; then there exists a rule $f_A \leftarrow not\, f_A, A$ in P which must be satisfied by N. But since $N(A) = T$,

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

and N is solid, the only way to satisfy this rule is when $N(f_A) = T$. By definition of N', $N'(f_A) = T^*$. Therefore, $N' \prec N$. It is also easy to see that N' is a model of $P_1 \cup P$ because it satisfies all rules $f_A \leftarrow \mathit{not}\, f_A, A$ and f_A does not occur in any other rule of $P_1 \cup P$.

Therefore, M' is a most-preferred answer set of $P_1 \cup P$. This contradicts the assumption of strong equivalence because M' is not even a model for $P_2 \cup P$.

Case 2: M' is a model of P_2. Let D be an atom in $\Sigma - (\mathcal{T} \cup \mathcal{F})$ that does not occur in $P_1 \cup P_2$. Such an atom always exists, since Σ is a countably infinite set and $\mathcal{T}$, $\mathcal{F}$, P_1, and P_2 are finite; moreover, $M(D) = F$, by our assumption about M. We take

$$\begin{aligned} P = &\{A \leftarrow \mid M(A) = T\} \cup \\ &\{A \times t_A \leftarrow \mid t_A \in \mathcal{T}\} \cup \{f_A \leftarrow \mathit{not}\, f_A, A \mid f_A \in \mathcal{F}\} \cup \\ &\{B \leftarrow A \mid A \neq B \text{ and } M(A) = M(B) = T^*\} \cup \\ &\{D \leftarrow \mathit{not}\, A \mid M(A) = T^*\} \end{aligned}.$$

It is easy to see that M' satisfies every formula in P, and therefore it is a model of both $P_1 \cup P$ and $P_2 \cup P$. We will show that M' is a most-preferred answer set of $P_2 \cup P$ but not a most-preferred answer set of $P_1 \cup P$.

We proceed by showing that M' is a $\preceq$-minimal model of $P_2 \cup P$. Assume there exists a model N of $P_2 \cup P$ such that $N \prec M'$.

We first show that there exists an atom A such that $M(A) = T^*$ and $N(A) = T$. Consider an arbitrary atom C. If $M(C) = T$, then it is also $N(C) = T$, because P contains $C \leftarrow$ and N is a model of P. If $M(C) = F^*$, then by the construction of M' it is $M'(C) = F^*$. Since N is a model of P, by property P1 we obtain $N(C) \neq F$. This implies $N(C) = F^*$, because $N \prec M'$. If $M(C) = F$, then by the construction of M' it is $M'(C) = F$, and since $N \prec M'$ we get $N(C) = F$. Therefore, if $M(C) \neq T^*$, then $M(C) = N(C)$. There should be, however, an atom A that occurs in P_2 such that $N(A) \neq M(A)$ because N is a model of P_2 and M is not. Obviously, for that atom it must be $M(A) = T^*$ and $N(A) \neq T^*$. Now, notice that there exists a rule $D \leftarrow \mathit{not}\, A$ in P where $M(D) = F$ and must be satisfied by N since it is also a model of P. Since $M(D) = F$ implies $N(D) = F$, the only remaining possibility is $N(A) = T$.

We next show that there exists an atom B such that $M(B) = N(B) = T^*$. Since $N \prec M'$, there exists B such that $N(B) \prec M'(B)$. The last relation immediately implies $M'(B) \neq F$. Notice also that, by the construction of M', it is $M'(B) \neq T^*$. Moreover, it cannot be $M'(B) = F^*$, since in that case from $N(B) \prec M'(B)$ we would obtain $N(B) = F$, which contradicts property P1. Therefore, the only remaining value is $M'(B) = T$. For that atom, it cannot be $M(B) = T$ because then it would also be $N(B) = T$ (since $B \leftarrow$ is a rule in P and N is a model of P), which would contradict $N(B) \prec M'(B)$. It follows by the construction of M' that $M(B) = T^*$. We claim that $N(B) = T^*$, that is, it cannot be $N(B) = F$. Since $M(B) = T^*$ there exists a rule $D \leftarrow \mathit{not}\, B$ where $M(D) = F$. Since, $M(D) = F$, we get $N(D) = F$. If we assume that also $N(B) = F$ then N does not satisfy this rule which is a contradiction. Therefore, $N(B) = T^*$.

Since $M(A) = M(B) = T^*$ there exists rule $B \leftarrow A$ in P that is not satisfied by N because we have showed that $N(B) = T^*$ and $N(A) = T$. Therefore, N is not a model of $P_2 \cup P$, which is a contradiction.

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

We conclude that M' is $\preceq$-minimal model of $P_2 \cup P$. Following an identical reasoning as in the final paragraph of the proof of Case 1, we can show that M' is a most-preferred answer set of $P_2 \cup P$. In order to conclude the proof, it suffices to show that M' is not a most-preferred answer set of $P_1 \cup P$. We define M'' as:

$$M''(A) = \begin{cases} T & A \in \mathcal{T} \\ F^* & A \in \mathcal{F} \\ M(A) & \text{otherwise} \end{cases}.$$

M'' is not a model of P_2 because M is not a model of P_2. By definition, $M'' \preceq M'$. But $M'' \neq M'$ because M' is a model of P_2 and M'' is not. Therefore, $M'' \prec M'$. Observe that M'' agrees with M for all A that appear in P_1 and since M is a model of P_1, M'' is also a model of P_1. M'' also satisfies the rules of P, and therefore it is a model of $P_1 \cup P$. Therefore, M' is not a most-preferred answer set of $P_1 \cup P$. □

We now consider the second notion of strong equivalence that is applicable in our setting.

Definition 11
Two LPODs P_1, P_2 are termed *strongly equivalent under all the answer sets*, if for every LPOD P, $P_1 \cup P$ and $P_2 \cup P$ have the same answer sets.

Theorem 2
Two LPODs P_1, P_2 are strongly equivalent under all the answer sets if and only if they are logically equivalent in four-valued logic.

The proof of the above theorem follows the same steps as that of the proof of Theorem 1, omitting the parts of the proof related to $\sqsubset$-minimization.

Corollary 1
Two LPODs P_1, P_2 are strongly equivalent under the most-preferred answer sets if and only if they are strongly equivalent under all the answer sets.

We feel that the above corollary highlights an interesting fact: it states that assessing the observable behaviour of two programs with respect to the most-preferred answer sets, suffices to determine strong equivalence of the programs.

Due to the above corollary, in the following we will often talk about "strong equivalence of LPODs" without specifying the exact type of equivalence (since they coincide).

Example 2
One can easily verify (using a four-valued truth table or a case analysis) that the programs:

$$\begin{array}{l} \mathtt{a} \times \mathtt{b} \leftarrow \\ \mathtt{a} \leftarrow \end{array},$$

and the program that consists of just the following fact:

$$\mathtt{a} \leftarrow,$$

are strongly equivalent. Similarly, one can verify that the programs given in Example 3 of the paper by Faber *et al.* (2008), namely:

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

```
c × a × b ←
a ← c
b ← c          ,
c ← a,b
```

and:

```
c × a × b ←
c × c × b × a ←
a ← c                ,
b ← c
c ← a,b
```

are also strongly equivalent. Notice that the above two programs are also strongly equivalent under the relations $\equiv^{i}_{s,\times}$ and $\equiv_{s,\times}$ defined in the paper by Faber *et al.* (2008) (see the discussion in Example 3, page 441, of the aforementioned paper).

We now demonstrate that our characterization of strong equivalence, when restricted to normal logic programs, retains the spirit of the initial characterization of strong equivalence for such programs [2] (Lifschitz *et al.* 2001). More specifically, we show that in order to characterize strong equivalence for normal logic programs, it suffices to look at their models that contain only the truth values F, T^*, and T.

We define strong equivalence for normal programs in the standard way (Lifschitz *et al.* 2001). The "standard answer set semantics" is the usual stable model semantics (Gelfond and Lifschitz 1988) of normal logic programs.

Definition 12
Two normal logic programs P_1 and P_2 are termed *strongly equivalent under the standard answer set semantics*, if for every normal logic program P, $P_1 \cup P$ and $P_2 \cup P$ have the same standard answer sets.

The following definition and theorem characterize strong equivalence of normal programs in our setting.

Definition 13
An interpretation I of an LPOD P is called *three-valued* if for all atoms A in P, it is $I(A) \neq F^*$. A *three-valued model* of P is a three-valued interpretation of P that is also a model of P.

Theorem 3
Let P_1 and P_2 be normal logic programs. Then, P_1 and P_2 are strongly equivalent under the standard answer set semantics if and only if they have the same three-valued models.

4 The complexity of strong equivalence for LPODs

In this section we examine the complexity of strong equivalence under our new characterization. Since the two versions of strong equivalence that we have examined have an

[2] Actually, the syntax of the programs treated in the paper by Lifschitz *et al.* (2001), is broader than that of normal logic programs.

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

identical characterization (see Theorems 1 and 2), the same complexity applies in both cases.

Our proof establishes **coNP**-hardness by a direct (and quite simple) reduction from 3SAT, which uses the special structure of LPODs in a crucial way. The corresponding proof by Faber *et al.* (2008) utilizes the more involved **coNP**-hardness result of Lin (2002) for strong equivalence of normal logic programs [3]. In this respect, we feel that the proof that follows, apart from the fact that it applies to our new characterization of strong equivalence, also has an interest in its own right due to its different approach.

Theorem 4
Strong equivalence of LPODs is a **coNP**-complete problem.

Proof
Let P_1, P_2 be two LPODs that are not strongly equivalent. Then, without loss of generality, there exists a four-valued interpretation I that is a model of P_1, but not a model of P_2. Assume that the ground atoms that occur in $P_1 \cup P_2$ are $A_1, A_2, \ldots, A_m$, and consider the certificate $\mathcal{C} = [A_1, I(A_1), A_2, I(A_2), \ldots, A_m, I(A_m)]$. $\mathcal{C}$ has size polynomial to the size of (P_1, P_2); moreover, given P_1, P_2, and $\mathcal{C}$, it can be verified in polynomial time that P_1 and P_2 are not strongly equivalent. Thus, deciding whether two programs are strongly equivalent is in **coNP**.

We next prove that strong equivalence of LPODs is also a **coNP**-hard problem, using a polynomial time reduction of 3SAT to the complement of this problem.

Let $\phi = \bigwedge_{i=1}^{n} c_i$ be a propositional formula in conjunctive normal form, where $c_i = L_{i,1} \vee L_{i,2} \vee L_{i,3}$ and $L_{i,j}$ is a literal (i.e. either a variable or the negation of a variable). For convenience, we may assume that the variables that occur in ϕ are elements of Σ.

We will construct two programs P_1, P_2, such that ϕ is satisfiable if and only if P_1 and P_2 are not strongly equivalent.

For every literal L we define $\widetilde{L}$ as follows:

$$\widetilde{L} = \begin{cases} L & \text{if } L = C, \text{ for some } C \in \Sigma \\ not\ C & \text{if } L = \neg C, \text{ for some } C \in \Sigma \end{cases}.$$

Let A, B be two propositional variables in Σ that do not occur in ϕ and let Q be

$$Q = \{A \leftarrow \widetilde{L_{i,1}}, \widetilde{L_{i,2}}, \widetilde{L_{i,3}} \mid 1 \leq i \leq n\}.$$

The LPODs P_1 and P_2 are defined as follows:

$$P_1 = Q \cup \{A \times B \leftarrow\},$$

$$P_2 = Q \cup \{A \times B \leftarrow\} \cup \{A \leftarrow\}.$$

Assume that ϕ is satisfiable and let J be a two-valued interpretation such that $J(\phi) = T$. We define the four-valued interpretation I as follows:

$$I(A) = F^*$$
$$I(B) = T$$

[3] As remarked by one of the reviewers, the **coNP**-completeness of strong equivalence for standard ASP programs was first shown in the paper by Pearce *et al.* (2001).

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

$$I(C) = F, \text{ if } C \text{ occurs in } \phi \text{ and } J(C) = T$$
$$I(C) = T, \text{ if } C \text{ occurs in } \phi \text{ and } J(C) = F.$$

Consider an arbitrary rule $A \leftarrow \widetilde{L_{i,1}}, \widetilde{L_{i,2}}, \widetilde{L_{i,3}}$ in Q. Then, $L_{i,1} \vee L_{i,2} \vee L_{i,3}$ is a clause in ϕ; since J satisfies ϕ, it holds $J(L_{i,j}) = T$, for some $j \in \{1,2,3\}$. Therefore, $I(\widetilde{L_{i,j}}) = F$, which implies that I satisfies the rule $A \leftarrow \widetilde{L_{i,1}}, \widetilde{L_{i,2}}, \widetilde{L_{i,3}}$. Moreover, $I(A \times B) = T$. We conclude that I is a model of P_1; however, I is not a model of P_2, since $I(A) = F^*$. Therefore, P_1 and P_2 are not logically equivalent in the four-valued logic, which implies that they are not strongly equivalent.

Conversely, assume that P_1 and P_2 are not strongly equivalent. Then, P_1 and P_2 are not logically equivalent in our four-valued logic. Since $P_1 \subset P_2$, there exists a four-valued interpretation I that is a model of P_1, but not a model of P_2. We define the following two-valued interpretation for the variables in ϕ:

$$J(C) = \begin{cases} T & \text{if } I(C) \leq F^* \\ F & \text{if } I(C) \geq T^* \end{cases}.$$

We will show that J satisfies ϕ. We first prove some properties of I.

Since I is a model of P_1, it must be either $I(A) = T$, or $I(A) = F^*$ and $I(B) = T$, so that the rule $A \times B \leftarrow$ is satisfied. However, in the former case, I should also be a model of P_2 (since $P_2 - P_1 = \{A \leftarrow\}$), which is a contradiction. Therefore, only the latter case is possible, that is, $I(A) = F^*$.

Consider an arbitrary clause $c_i = L_{i,1} \vee L_{i,2} \vee L_{i,3}$ in ϕ. Since I is a model of P_1, I satisfies the rule $A \leftarrow \widetilde{L_{i,1}}, \widetilde{L_{i,2}}, \widetilde{L_{i,3}}$ in $Q \subset P_1$. Therefore, $\min\{I(\widetilde{L_{i,1}}), I(\widetilde{L_{i,2}}), I(\widetilde{L_{i,3}})\} \leq I(A) = F^*$, which implies that there exists a $j \in \{1,2,3\}$ such that $I(\widetilde{L_{i,j}}) \leq F^*$. But then, $J(L_{i,j}) = T$. We conclude that ϕ is satisfiable. □

5 Related and future work

The work on strong equivalence, started with the pioneering results of Lifschitz *et al.* (2001), but has since been extended to various formal systems. In particular, strong equivalence has been abstractly studied as a property across a variety of preferential formalisms (Faber *et al.* 2013). To our knowledge however, the only existing work on the strong equivalence of LPODS is the paper by Faber *et al.* (2008). In that work the authors present an exhaustive study of several notions of strong equivalence for LPODs. More specifically, given LPODs P, Q, they consider the following notions of strong equivalence:

1. $P \equiv_s Q$ holds iff the standard answer sets of P and Q coincide under any extension by ordinary (namely, *normal*) programs.
2. $P \equiv_{s,\times} Q$ holds iff the standard answer sets of P and Q coincide under any extension by LPODs.
3. $P \equiv_s^\sigma Q$ holds iff the σ-preferred answer sets of P and Q coincide under any extension by ordinary programs, where $\sigma \in \{i, p, c\}$ and the indices i, c, and p correspond to the *inclusion*, *Pareto*, and *cardinality* orderings, respectively (see the paper by Brewka *et al.* (2004b) for formal definitions of these orderings).
4. $P \equiv_{s,\times}^\sigma Q$ holds iff the σ-preferred answer sets of P and Q coincide under any extension by LPODs, where $\sigma \in \{i, p, c\}$.

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

Considering the above notions, the study of Faber *et al.* (2008) is certainly broader than the present work. We have not considered cases (1) and (3) above because in the standard definition of strong equivalence (Lifschitz *et al.* 2001) both the programs under comparison and the context programs, all belong to the same source language (in our case, LPODs). Of course, there may exist application domains where relations like $\equiv_s$ and $\equiv_s^\sigma$ might be of interest. In such a case, it might prove interesting to extend the present work in this direction. Case (2) above is covered by our Theorem 2. Finally, from case (4) above, we cover only the subcase where σ is the *inclusion* preference. The subcases of Pareto and cardinality preferences are not covered because the semantics of Charalambidis *et al.* (2021) on which the present work is based, is defined using the relation $\sqsubset$, which is the model-theoretic version of the inclusion preference of Brewka (2002) and Brewka *et al.* (2004b). It is important, however, to stress that the inclusion preference is probably the most fundamental among the three orderings and the initial paper introducing LPODs (Brewka 2002), used only this one. The Pareto and cardinality preferences were proposed subsequently in order to remedy the shortcomings of the initial semantics of LPODs (Brewka *et al.* 2004b, see the discussion in page 342). Notice also that the cardinality preference can not be generalized in a direct way to first-order programs whose ground instantiation consists of an infinite number of rules.

Recapitulating, the two notions of strong equivalence that we cover in the present paper (Theorems 1 and 2), correspond to the relations $\equiv^i_{s,\times}$ and $\equiv_{s,\times}$ defined in the paper by Faber *et al.* (2008). In our case, both notions of strong equivalence coincide, because they have a unique characterization as logical equivalence in our four-valued logic. On the other hand, the relations $\equiv^i_{s,\times}$ and $\equiv_{s,\times}$ do not coincide (Faber *et al.* 2008, see Theorem 21). This means that our approach and that of Faber *et al.* (2008) are different: there exist programs that are strongly equivalent with respect to one of the approaches and not strongly equivalent with respect to the other approach. This was expected since the two approaches are based on markedly different semantics. Although it does not seem straightforward to establish a formal relation between our framework and that of Faber *et al.* (2008), we can find examples where the two approaches give different results.

Example 3
Consider the following two programs given in Example 2 of the paper by Faber *et al.* (2008):

```
c × a × b ←
c ← a,b      ,
d ← c,not d
```

and:

```
c × b × a ←
c ← a,b      .
d ← c,not d
```

It is intuitively clear that in the first program `a` is preferred over `b`, while in the second program `b` is preferred over `a`. Despite this difference, the two programs are strongly equivalent under the $\equiv_{s,\times}$ semantics of Faber *et al.* (2008). Under our characterization the two programs are not strongly equivalent. To see this, consider the interpretation $I = \{(\mathtt{a}, T), (\mathtt{b}, F), (\mathtt{c}, F^*), (\mathtt{d}, F^*)\}$, which is a model of the first program but not a model of the second. Therefore, the two programs are not logically equivalent in our four-valued logic, and consequently they are not strongly equivalent in our setting.

https://doi.org/10.1017/S1471068422000242 Published online by Cambridge University Press

Although our study does not cover all the notions of strong equivalence examined in Faber *et al.* (2008), we believe that it has important advantages. Our work characterizes strong equivalence as logical equivalence in the four-valued logic of Charalambidis *et al.* (2021). This result extends in a smooth way the well-known characterization of strong equivalence for normal logic programs (Lifschitz *et al.* 2001). Notice that the corresponding characterization of the inclusion preferred strong equivalence in Faber *et al.* (2008) is much more involved and uses certain binary functions over the sets of models of the programs that rely on the syntax of the given programs (see Faber *et al.* (2008), Definition 8 and Theorem 19). We believe that this is not an inherent shortcoming of the work of Faber *et al.* (2008), but instead a possibly unavoidable consequence of the fact that the original semantics of LPODs (Brewka 2002; Brewka *et al.* 2004b) is not purely model-theoretic. The simplicity of our characterization makes us believe that it can be extended to broader classes of programs, such as for example to LPODs with strong negation and to disjunctive LPODs (Charalambidis *et al.* 2021).

One important aspect that we have not examined in this paper, is the possible practical use of the proposed strong equivalence characterization. To our knowledge, all major ASP systems are two-valued, and it is therefore a legitimate question of how our four-valued framework can be embedded in such systems. We believe that a promising direction for future work would be to define a notion of *collapsed strong equivalence* for LPODs:

Definition 14
Two LPODs P_1 and P_2 are termed *collapsed strongly equivalent under the most-preferred answer sets* if for every LPOD P, the most-preferred answer sets of $P_1 \cup P$ and $P_2 \cup P$ become identical when F^* is collapsed to F.

Notice that in the above definition we do not need to collapse T^* to T because, by Definition 7, answer sets do not contain the T^* value. The logical characterization of collapsed strongly equivalent LPODs is probably an interesting question that deserves further investigation.

Finally, a very interesting question raised by one of the reviewers is whether the techniques developed in the paper by Charalambidis *et al.* (2021) can also be used to derive a novel and simpler semantics for Qualitative Choice Logic (QCL) (Brewka *et al.* 2004a). Notice that QCL has also recently been investigated with respect to strong equivalence (Bernreiter *et al.* 2021), so the work developed in the present paper may be also relevant in this more general context.

Supplementary material

To view supplementary material for this article, please visit http://doi.org/10.1017/S1471068422000242.

References

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'03 Workshop, Messina, Italy, September 26-28, 2003*, vol. 78. *CEUR Workshop Proceedings*. CEUR-WS.org.

Bernreiter, M., Maly, J. and Woltran, S. 2021. Choice logics and their computational properties. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event/Montreal, Canada, 19-27 August 2021,* Z. Zhou, Ed. ijcai.org, 1794–1800.

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* 2002. AAAI Press/The MIT Press, 100–105.

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

Brewka, G., Niemelä, I. and Syrjänen, T. 2004b. 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.

Charalambidis, A., Rondogiannis, P. and Troumpoukis, A. 2021. A logical characterization of the preferred models of logic programs with ordered disjunction. *Theory and Practice of Logic Programming*, *21*, 5, 629645.

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.

Faber, W., Truszczynski, M. and Woltran, S. 2013. Abstract preference frameworks - a unifying perspective on separability and strong equivalence. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*, M. desJardins and M. L. Littman, Eds. AAAI Press.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, USA, August 15–19, 1988 (2 Volumes)*. MIT Press, 1070–1080.

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

Lin, F. 2002. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proceedings of the Eights International Conference on Principles and Knowledge Representation and Reasoning (KR-02), Toulouse, France, April 22-25, 2002*, D. Fensel, F. Giunchiglia, D. L. McGuinness and M. Williams, Eds. Morgan Kaufmann, 170–176.

Pearce, D. 1966. A new logical characterisation of stable models and answer sets. In *Non-Monotonic Extensions of Logic Programming, NMELP '96, Bad Honnef, Germany, September 5-6, 1996, Selected Papers*, vol. 1216. *Lecture Notes in Computer Science*. Springer, 57–70.

Pearce, D. 1999. *From Here to There: Stable Negation in Logic Programming*. Applied Logic Series. Springer Netherlands, Dordrecht, 161–181.

Pearce, D., Tompits, H. and Woltran, S. 2001. Encodings for equilibrium logic and logic programs with nested expressions. In *Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving, 10th Portuguese Conference on Artificial Intelligence, EPIA 2001, Porto, Portugal, December 17-20, 2001, Proceedings*, P. Brazdil and A. Jorge, Eds., vol. 2258. *Lecture Notes in Computer Science*. Springer, 306–320.