Explaining Preferences, Preferring Explanations*

Pedro Cabalar and Jorge Fandinno

Department of Computer Science
University of Corunna, Spain
{cabalar, jorge.fandino}@udc.es

Abstract. In this paper we study the possibility of providing causal explanations for preferred answer sets, as those obtained from logic programs with ordered disjunction (LPODs). We use a recently defined multi-valued semantics for answer sets based on a causal algebra and consider its direct application to LOPDs by several illustrating examples. We also explain the limitations of this simple approach and enumerate some open topics to explore in the future.

1 Introduction

Although much work in problem solving has been devoted to problems with a small set of solutions that are hard to find, there are many other situations in which the number of solutions is astronomical, but not all of them are preferred in the same way. Think, for instance, on the configuration of a timetable for a university school: there exists a huge number of combinations with physically feasible timetables, but most of them do not have a *reasonable* distribution.

The definition of a suitable Knowledge Representation (KR) language for specifying preferences has proved to be a difficult endeavour. Apart from the long list of features usually expected from a KR formalism (simplicity, clear semantics, flexibility, elaboration tolerance, computability, complexity assessment, efficient inference methods, etc), the specification of preferences has an extra difficulty that has to do with their *subjective, ambiguous definition*. For instance, while any student or teacher could easily tell why a given random timetable is not reasonable, it is very difficult to formally encode all the preferences that capture the commonsense idea of "*reasonable timetable*" in the general case – furthermore, there would not be a complete agreement among different persons either. Even when we have a single person, she may initially declare a list of preferences such as "*A* better than *B*." However, when this list grows up, the results obtained by formal systems are usually far away from the expected outcomes that the person had in mind. To bridge the gap, several refinements can be applied, such as adding conditional preferences or including an ordering among them. Still, while the strict physical rules that govern a timetable assignment are objective and accurate, it is practically impossible to guarantee that the set of preferences is working *as expected* from a commonsense point of view.

Although there exist both qualitative and quantitative approaches for dealing with preferences in Artificial Intelligence, the interest in KR has been mostly focused on

* This research was partially supported by Spanish MEC project TIN2013-42149-P

the qualitative orientation, probably because it is closer to commonsense reasoning, as humans rarely express their preferences in numerical terms[1].

As explained in [7], the relation between preferences and Non-Monotonic Reasoning (NMR) has been evident from the very beginning. On the one hand, the nature of preferences is clearly non-monotonic: the addition of new preferences may drastically change the obtained conclusions. On the other hand, we can also see a default as a kind of preference specification: a sentence like "birds normally fly" can be read as "if X is a bird, my *preferred belief* is that X flies." Indeed, many non-monotonic formalisms are defined in terms of a preference relation among logical models of a theory.

When one mentions research on preferences and NMR there is one researcher's name that immediately comes to mind: *Gerhard Brewka*. Being one of the pioneers in NMR, Gerd soon became interested in the topic of preferences, proposing extensions of Reiter's Default Logic [21] to include priorities among defaults [1, 2]. He also got interested on an emerging problem solving paradigm, *Answer Set Programming* (ASP) [17, 16] whose semantics (*stable models* [14] or *answer sets*) can also be seen as a particular case of Default Logic. ASP has become nowadays [5] a de facto standard for practical NMR and problem solving thanks to its clear semantics and the availability of efficient solvers together with a wide spectrum of applications running in real scenarios. In 1998, Gerd coauthored, together with Thomas Eiter, one of the first remarkable approaches of preferences in ASP [4]. Four years later, he introduced a different orientation called *Logic Programs with Ordered Disjunction* (LOPDs) [3]. In LPODs, logic programs were extended with a new connective called *ordered disjunction* allowing the representation of ranked options for problem solutions in the heads of rules. As a result, LPODs provided a flexible and readable way for expressing (conditional) preferences combined with the expressiveness of default negation already embodied in ASP. Originally, the semantics of LPODs relied on an adapted definition of answer sets or, alternatively, resorted to program transformations (so-called "split programs"). However, in [8], it was shown how the ordered disjunction connective could be naturally captured in *Equilibrium Logic* [18], the most general logical characterisation of ASP. This actually allows seeing LPODs as a "regular" type of ASP programs extended with an additional preference relation on answer sets.

As we explained before, in a practical scenario, one can expect that the specification of preferences is obtained after several attempts and refinements, by repeatedly observing the obtained results and comparing them to the expected behaviour. In such a context, it seems clear that explanations play a fundamental role. There exist several approaches in the ASP literature focused on providing explanations for debugging [13, 20, 22, 11] or obtaining justifications for the program outcome [19, 12, 24]. In a recent proposal [9], the idea of *causal justifications* for ASP programs was introduced. These causal justifications are embodied in ASP as a multi-valued extension where each true atom in a stable model is associated an expression involving rule labels corresponding to the alternative proofs to derive the atom.

[1] An exception is, perhaps, when we consider optimization problems (minimizing cost, maximizing profit, etc) as an instance of preference specification. In any case, we mean here that, even though we are sometimes able to assign numerical weights to preferences, this is not usually present at our commonsense level, but a forced assignment *a posteriori*.

In this paper we study the possibility of providing explanations for LPODs. As a first direct attempt, we have considered the combination of LPODs with causal justifications, showing its behaviour on several examples. We also explain how it may seem sometimes reasonable to provide preferences, not only among program rules, but also on the explanations obtained. Finally, we discuss the obvious limitations of this first approach and foresee some interesting open topics to explore in the future.

2 Logic Programs with Ordered Disjunction

2.1 Preliminaries

We begin introducing some preliminary notation that will be useful later. Let $\mathbf{A}$ be a (possibly empty) list of (possibly repeated) formulas. We write $|\mathbf{A}|$ to stand for the length of $\mathbf{A}$. For any $k \in \{1,\ldots,|\mathbf{A}|\}$, by $\mathbf{A}[k]$ we mean the k-th expression in $\mathbf{A}$ and by $\mathbf{A}[1..k]$, the prefix of $\mathbf{A}$ of length k, that is, $\mathbf{A}[1]\ldots\mathbf{A}[k]$. For a binary operator $\odot \in \{\vee,\wedge,\times\}$, by $(\odot\mathbf{A})$ we mean the formula resulting from the repeated application of $\odot$ to all formulas in $\mathbf{A}$ in the same ordering. As an example, given the sequence of atoms $\mathbf{A} = (a,b,c,d,e)$, the expression $(\times\mathbf{A}[1..3])$ represents the formula $a \times b \times c$. We write *not* $\mathbf{A}$ to stand for the sequence of formulas $(\mathit{not}\ \mathbf{A}[1])\ldots(\mathit{not}\ \mathbf{A}[k])$ where $k = |\mathbf{A}|$. An empty conjunction is understood as $\top$ whereas an empty disjunction (both ordered $\times$ or regular $\vee$) is understood as $\bot$. The concatenation of two lists of formulas, $\mathbf{A}$ and $\mathbf{B}$, is simply written as $\mathbf{AB}$.

A *logic program* is a set of rules of the form:

$$(\vee\mathbf{A}) \vee (\vee \mathit{not}\ \mathbf{A}') \leftarrow (\wedge\mathbf{B}) \wedge (\wedge \mathit{not}\ \mathbf{B}') \tag{1}$$

where $\mathbf{A},\mathbf{A}',\mathbf{B}$ and $\mathbf{B}'$ are lists of atoms. The consequent and antecedent of the implication above are respectively called the *head* and the *body* of the rule. In the examples, we will usually represent conjunctions in the body as commas, to follow the standard ASP notation. A rule with an empty head $\bot$ (that is, $|\mathbf{A}| + |\mathbf{A}'| = 0$) is called a *constraint*. A rule with an empty body $\top$ (that is, $|\mathbf{B}| + |\mathbf{B}'| = 0$) is called a *fact*, and we usually write the head F instead of $F \leftarrow \top$. A rule is said to be *normal* when $|\mathbf{A}| = 1$ and $|\mathbf{A}'| = 0$. A rule is *positive* when $|\mathbf{A}'| = |\mathbf{B}'| = 0$. We extend the use of these adjectives to a program, meaning that all its rules are of the same kind.

A rule head of the form $p \vee \neg p$ behaves as a choice rule: we are free to include p or not. This kind of head is usually written as $\{p\}$ in ASP and we will sometimes use it too to increase readability.

Answer sets of a program P are defined in terms of the classical Gelfond-Lifschitz's reduct [14], that is extended as follows for the syntactic case we are considering (disjunctive heads with default negation [**?**]). The *reduct* of a program P with respect to a set of atoms I, written P^I, consists of a rule of the form $(\vee\mathbf{A}) \leftarrow (\wedge\mathbf{B})$ per each rule in P of the form (1) that satisfies $I \models (\wedge\mathbf{A}') \wedge (\wedge \mathit{not}\ \mathbf{B}')$. We say that a set of atoms I is an *answer set* of a program P if I is a minimal model of P^I.

Answer sets differ from stable models in that they further allow a new negation (called *classical*, *explicit* or *strong*). For simplicity, we will understand strong negation

of an atom p as a new atom "$\neg p$" and assume that each time one of these "negative" atoms is used, we implicitly further include the constraint:

$$\bot \leftarrow p, \neg p$$

2.2 LPODs: original definition

A *logic program with ordered disjunction* (LPOD) is a set of rules of the form:

$$(\times \mathbf{A}) \leftarrow (\wedge \mathbf{B}) \wedge (\wedge \mathit{not}\ \mathbf{B}') \tag{2}$$

where $\mathbf{A}, \mathbf{B}$ and $\mathbf{B}'$ are lists of atoms. We say that a set of atoms I *satisfies* an LPOD rule r such as (2), written $I \models r$, when $I \models (\vee \mathbf{A}) \leftarrow (\wedge \mathbf{B}) \wedge (\wedge \mathit{not}\ \mathbf{B}')$ in classical logic.

For each LPOD rule r of the form (2), we define its k-th *option*, written r_k, with $k \in \{1, \ldots, |\mathbf{A}|\}$, as the normal rule:

$$\mathbf{A}[k] \leftarrow (\wedge \mathbf{B}) \wedge (\wedge \mathit{not}\ \mathbf{B}') \wedge (\wedge \mathit{not}\ \mathbf{A}[1..k{-}1])$$

A normal logic program P' is a *split program* of P if it is the result of replacing each LPOD rule $r \in P$ by one of its possible options r_k. A set of atoms I is an *answer set* of P if it is an answer set of some split program P' of P.

Example 1 (From [6]). Let P_1 be the LPOD:

$$a \times b \leftarrow \mathit{not}\ c \qquad\qquad b \times c \leftarrow \mathit{not}\ d$$

This LPOD has four split programs:

$a \leftarrow \mathit{not}\ c$ $b \leftarrow \mathit{not}\ d$	$a \leftarrow \mathit{not}\ c$ $c \leftarrow \mathit{not}\ d, \mathit{not}\ b$
$b \leftarrow \mathit{not}\ c, \mathit{not}\ a$ $b \leftarrow \mathit{not}\ d$	$b \leftarrow \mathit{not}\ c, \mathit{not}\ a$ $c \leftarrow \mathit{not}\ d, \mathit{not}\ b$

that yield three answer sets $\{a,b\}$, $\{c\}$ and $\{b\}$. □

As explained in [6], answer sets of LPODs can also be described in terms of a program reduct, instead of using split programs.

Definition 1 ($\times$-reduct). *The* $\times$-reduct *of an LPOD rule r such as* (2) *with respect to a set of atoms I denoted as $r^I_\times$ and defined as the set of rules:*

$$\mathbf{A}[i] \leftarrow (\wedge \mathbf{B}) \tag{3}$$

for all $i = 1, \ldots, |\mathbf{A}|$ such that $I \models (\wedge \text{not } \mathbf{B}') \wedge (\wedge \text{not } \mathbf{A}[1..i{-}1]) \wedge \mathbf{A}[i]$. □

As expected, the $\times$-*reduct* of an LPOD P with respect to I, written $P^I_\times$ is the union of all $r^I_\times$ for all LPOD rules $r \in P$. For instance, for $I = \{b, c\}$ and P:

$$a \times b \leftarrow c, not\ d \tag{4}$$

$$d \times a \leftarrow not\ b \tag{5}$$

$$d \times e \leftarrow not\ a \tag{6}$$

the reduct $P^I_\times$ would be the rule $\{b \leftarrow c\}$. Notice that $P^I_\times$ defined in this way is always a positive (non-disjunctive) logic program and so it has a least model [23].

Theorem 1 (From [6]). *A set of atoms I is an answer set of an LPOD P iff $I \models P$ and I is the least model of $P^I_\times$.* □

It is important to note that $I \models P^I_\times$ does not imply $I \models P$, and thus, the latter is also required in the above theorem. For instance, in the last example, the interpretation $\emptyset$ is the least model of $P^I_\times$ but does not satisfy the LPOD rule (6).

Three ordering relations can be used for selecting preferred answer sets. We say that an LPOD rule r of the form (2) is *satisfied to degree* $j \in \{1, \ldots, |\mathbf{A}|\}$ by a set of atoms I, written $I \models_j r$, when: I does not satisfy the body of r and $j = 1$; I satisfies the body of r and j is the minimum index for which $\mathbf{A}[j] \in I$. We define $deg_I(r) \stackrel{\text{def}}{=} j$ when $I \models_j r$ and define the set $I^j(P) \stackrel{\text{def}}{=} \{r \in P \mid I \models_j r\}$. Given two answer sets I, J of a given LPOD:

1. I is *cardinality-preferred* to J, written $I >_c J$, when for some truth degree k, $|I^k(P)| > |J^k(P)|$ whereas $|I^i(P)| = |J^i(P)|$ for all $i < k$.
2. I is *inclusion-preferred* to J, written $I >_i J$, when for some truth degree k, $J^k(P) \subset I^k(P)$ while $I^i(P) = J^i(P)$ for all $i < k$.
3. I is *Pareto-preferred* to J, written $I >_p J$, if for some rule $r \in P$, $deg_I(r) < deg_J(r)$ whereas for no rule $r' \in P$, $deg_I(r') > deg_J(r')$.

2.3 Ordered disjunction as an operator

As explained in the introduction, in [8] it was shown how the original definition of answer sets for LPODs can be alternatively characterised by a logical encoding into Equilibrium Logic [18], and in particular, into its monotonic basis called the logic of *here-and-there* (HT) [15]. [8] showed that the expression $A \times B$ can be defined in HT as $A \vee (not\ A \wedge B)$. As a result of some simple transformations in the logic of HT, it was possible to prove that a rule of the form (2) can be seen as an abbreviation of the conjunction of the following rules:

$$\mathbf{A}[k] \vee not\ \mathbf{A}[k] \leftarrow (\wedge \mathbf{B}) \wedge (\wedge not\ \mathbf{B}') \wedge (\wedge not\ \mathbf{A}[1..k-1]) \tag{7}$$

$$\bot \leftarrow (\wedge \mathbf{B}) \wedge (\wedge not\ \mathbf{B}') \wedge (\wedge not\ \mathbf{A}) \tag{8}$$

for all $k = 1, \ldots, |\mathbf{A}|$. For a rule r like (2) we denote each rule (7) by $r[k]$. As an example, the HT translation of the LPOD rule $r : a \times b \times c \leftarrow p, not\ q$ consists of:

$$\begin{aligned}
r[1]: &\ \ a \vee not\ a \leftarrow p, not\ q \\
r[2]: &\ \ b \vee not\ b \leftarrow p, not\ q, not\ a \\
r[3]: &\ \ c \vee not\ c \leftarrow p, not\ q, not\ a, not\ b \\
&\bot \leftarrow p, not\ q, not\ a, not\ b, not\ c
\end{aligned}$$

Note that rule heads have the form of choice expressions: as explained before, we would usually write $a \vee \neg a$ as $\{a\}$. So, essentially, $r[1]$ says that when $p \wedge not\ q$, we have freedom to choose a or not. In its turn, $r[2]$ further says that if, additionally, a is false, then we can freely choose b or not, and so on. The constraint just checks that at least one choice is eventually chosen.

At a first sight, this definition may seem similar to the one based on split programs. Note that, in fact, $r[k]$ can be seen as a weaker version of the previously defined r_k where we have just added *not* $\mathbf{A}[k]$ with a disjunction in the head. However, it is important to note that the HT translation provides a unique ASP program and that the translation of each LPOD rule (2) is modular, so we can safely understand it as an abbreviation of all the rules (7) and (8).

3 Causal justifications

In this section, we recall several definitions and notation from [9]. The intuitive idea is that atoms in a stable model will be assigned algebraic expressions instead of truth values $0, 1$. These algebraic expressions are built with labels for the program rules plus three operations: a product '$*$' meaning conjunction or joint causation; a concatenation '$\cdot$' meaning ordered sequence of application of terms; and an addition '$+$' meaning different alternative proofs for a same atom.

A *signature* is a pair $\langle At, Lb\rangle$ of sets that respectively represent *atoms* (or *propositions*) and rule *labels*.

The syntax is defined as follows. As usual, a *literal* is defined as an atom p (positive literal) or its default negation *not* p (negative literal). In this paper, we will concentrate on programs without disjunction in the head (leaving its treatment for future work).

Definition 2 (Causal logic program). *Given a signature $\langle At, Lb\rangle$, a* (causal) logic program *P is a set of rules of the form:*

$$t : H \vee (\vee \text{not } \mathbf{A}') \leftarrow (\wedge \mathbf{B}) \wedge (\wedge \text{not } \mathbf{B}') \tag{9}$$

where $t \in Lb \cup \{1\}$ where H is an atom and $\mathbf{A}, \mathbf{B}, \mathbf{B}'$ lists of atoms as before. □

For any rule R of the form (9) we define $label(R) \stackrel{\text{def}}{=} t$. When $t \in Lb$ we say that the rule is labelled; otherwise $t = 1$ and we omit both t and ':'. By these conventions, for instance, an unlabelled fact p is actually an abbreviation of $(1 : p \leftarrow)$. A logic program P is *positive* if it contains no default negation.

The semantics relies on assigning, to each atom, a causal term defined as follows.

Definition 3 (Causal term). *A* (causal) term*, t, over a set of labels Lb, is recursively defined as one of the following expressions $t ::= l \mid \prod S \mid \sum S \mid t_1 \cdot t_2 \mid (t_1)$ where $l \in Lb$, t_1, t_2 are in their turn causal terms and S is a (possibly empty and possible infinite) set of causal terms. When S is finite and non-empty, $S = \{t_1, \ldots, t_n\}$ we write $\prod S$ simply as $t_1 * \cdots * t_n$ and $\sum S$ as $t_1 + \cdots + t_n$. The set of causal terms is denoted by $\mathbf{T}_{Lb}$.* □

We assume that '$*$' has higher priority than '$+$'. When $S = \emptyset$, we denote $\prod S$ by 1 and $\sum S$ by 0. These values are the indentities for the product and the addition, respectively.

All three operations, '$*$', '$+$' and '$\cdot$' are associative. Furthermore, '$*$' and '$+$' are commutative and they hold the usual absorption and distributive laws with respect to infinite sums and products of any completely distributive lattice, as shown[2] in Figure 1. The behaviour of the '$\cdot$' operator is captured by the properties shown in Figure 2. As we can see, distributivity with respect to the product is only applicable to terms c, d, e without sums (this means that the empty sum, 0, is not allowed either). We define the standard order relation $\leq$ as follows:

$$t \leq u \qquad \text{iff} \qquad (t * u = t) \qquad \text{iff} \qquad (t + u = u)$$

By the identity properties of $+$ and $*$, this immediately means that 1 is the top element and 0 the bottom element of this order relation.

Associativity	*Commutativity*	*Absorption*
$t + (u+w) = (t+u) + w$	$t + u = u + t$	$t = t + (t * u)$
$t * (u*w) = (t*u) * w$	$t * u = u * t$	$t = t * (t+u)$

Distributive	*Identity*	*Idempotence*	*Annihilator*
$t + (u*w) = (t+u) * (t+w)$	$t = t + 0$	$t = t + t$	$1 = 1 + t$
$t * (u+w) = (t*u) + (t*w)$	$t = t * 1$	$t = t * t$	$0 = 0 * t$

Fig. 1. Sum and product satisfy the properties of a completely distributive lattice.

Absorption	*Associativity*	*Identity*	*Annihilator*
$t = t + u \cdot t \cdot w$	$t \cdot (u \cdot w) = (t \cdot u) \cdot w$	$t = 1 \cdot t$	$0 = t \cdot 0$
$u \cdot t \cdot w = t * u \cdot t \cdot w$		$t = t \cdot 1$	$0 = 0 \cdot t$

Indempotence	*Addition distributivity*	*Product distributivity*
$t \cdot t = t$	$t \cdot (u+w) = (t \cdot u) + (t \cdot w)$	$c \cdot d \cdot e = (c \cdot d) * (d \cdot e)$ with $d \neq 1$
	$(t+u) \cdot w = (t \cdot w) + (u \cdot w)$	$c \cdot (d * e) = (c \cdot d) * (c \cdot e)$
		$(c * d) \cdot e = (c \cdot e) * (d \cdot e)$

Fig. 2. Properties of the '$\cdot$' operator (c, d, e are terms without '$+$').

As proved in [9], any causal term can be equivalently reduced to a disjunctive normal form with an addition of products of pairs $(l \cdot l')$ where l, l' are labels. In fact, each product of pairs can be seen as a syntactic representation of a graph whose nodes are labels and with an arc (l, l') per each pair $(l \cdot l')$.

Given a signature $\langle At, Lb \rangle$ a *causal interpretation* is a mapping $I : At \rightarrow \mathbf{T}_{Lb}$ assigning a causal term to each atom. We denote the set of causal interpretations by $\mathbf{I}$.

[2] For readability sake, we only show the properties for finite sums and products, but they still hold in the infinite case.

For interpretations I and J we say that $I \leq J$ whether $I(p) \leq J(p)$ for each atom $p \in At$. Hence, there is a $\leq$-bottom interpretation $\mathbf{0}$ (resp. a $\leq$-top interpretation $\mathbf{1}$) that maps each atom p to 0 (resp. 1). Valuation of formulas is defined as follows:

$$I(not\ p) = \begin{cases} 1 \text{ if } I(p) = 0 \\ 0 \text{ otherwise} \end{cases} \qquad \begin{array}{l} I(\alpha \wedge \beta) = I(\alpha) * I(\beta) \\ I(\alpha \vee \beta) = I(\alpha) + I(\beta) \end{array}$$

An interpretation I *satisfies* a positive rule $t : H \leftarrow (\wedge \mathbf{B})$ when:

$$I(\wedge \mathbf{B}) \cdot t \leq I(H)$$

As usual, I is a *model* of a positive program iff it satisfies all its rules. Positive programs have a $\leq$-least model that corresponds to the least fixpoint of a direct consequences operator (see [9] for further details).

The *reduct* of a program P with respect to a causal interpretation I, written P^I, is defined in a similar manner as before. Program P^I consists of all positive rules $t : H \leftarrow (\wedge \mathbf{B})$ such that there is a rule (9) in P for which $I\big(\ (\wedge \mathbf{A}') \wedge (\wedge not\ \mathbf{B}')\big) = 1$.

Definition 4 (Causal model). *Given a positive causal logic program P, a causal interpretation I is a* causal stable model *iff I is the $\leq$-least model of P^I.* □

In [9], it was also shown that there exists a one-to-one correspondence between causal stable models of a program and its regular stable models (when labels are ignored). Furthermore, given the causal stable model I and its corresponding two-valued stable model J, the assignment $I(p)$ for an atom p precisely captures all the (non-redundant) proofs that can be built for deriving p in the positive program P^J.

4 Causal explanations for a preferred answer set

In this section we directly combine both approaches by extracting explanations of LPODs expressed as causal terms. Consider the following example from [3] about how to spend a free afternoon.

Example 2 (From [3]). You like to go to the beach, but also to the cinema. Normally you prefer the cinema over the beach, unless it is hot (which is the exception in the area where you live, except during the summer). If it is hot the beach is preferred over the cinema. In summer it is normally hot, but there are exceptions. If it rains the beach is out of question. □

This information can be captured by the following set of rules P_1:

$$\begin{array}{lrcl} c: & cinema \times beach & \leftarrow & not\ hot \\ b: & beach \times cinema & \leftarrow & hot \\ h: & hot & \leftarrow & not\ \neg hot, summer \\ r: & \neg beach & \leftarrow & rain \end{array}$$

This program has two choices with ordered disjunction, c and b, that respectively correspond to preferring *cinema* or *beach* depending on the context. As explained before, these rules can be unfolded into:

$$\begin{aligned}
c[1]: \{cinema\} &\leftarrow not\ hot \\
c[2]: \{beach\} &\leftarrow not\ hot, not\ cinema \\
\bot &\leftarrow not\ hot, not\ cinema, not\ beach \\
b[1]: \{beach\} &\leftarrow hot \\
b[2]: \{cinema\} &\leftarrow hot, not\ beach \\
\bot &\leftarrow hot, not\ beach, not\ cinema
\end{aligned}$$

Assume now we are given the information *summer*. The answer sets for $P_1 \cup \{summer\}$ are $J_0 = \{summer, hot, beach\}$ and $J_1 = \{summer, hot, cinema\}$ but only J_1 is preferred (under all preference orderings) since it satisfies rule b to degree 1 while J_2 only satisfies b to degree 2, and coincides in the rest of rules. As explained before, there exists one causal stable model per each regular stable model for any program P. In particular, the causal stable model I_1 corresponding to J_1 assigns the causal values $I_1(summer) = 1$, $I_1(hot) = h$, $I_1(beach) = h \cdot b[1]$ while all false atoms in J_1 are assigned the value 0. On the other hand, the causal stable model I_2 corresponding to J_2 only differs in the assignments $I(cinema) = h \cdot b[2]$ and $I(beach) = 0$. It is interesting to note that, since fact *summer* was not labelled, the truth value for that atom becomes 1 ("completely" true), which is the top element of the lattice of causal values. A second observation is that the causal value of atoms can also be used to find out the degree of satisfaction of an ordered choice rule. For instance, in I_1 we can see that rule b is being satisfied to degree 1 because the head atom in that position, *beach*, is assigned a value in which $b[1]$ occurs. Similarly, in I_2, we know that b is satisfied to degree 2 because $I_2(cinema)$ contains a reference to $b[2]$. If a program contains no repeated labels, it can be checked that, for any ordered choice rule r, we will never get two different occurrences $r[i]$ and $r[k]$ with $i \neq k$ in the values of atoms in a causal stable model.

Let us continue with the example as done in [3] and assume now that we add the information $\neg hot$. That is, take the program $P_1 \cup \{summer, \neg hot\}$. Then, the new preferred answer set becomes $J_3 = \{summer, \neg hot, cinema\}$ and its corresponding causal version I_3 makes the assignments $I_3(summer) = 1$, $I_3(\neg hot) = 1$ and $I_3(cinema) = c[1]$. A second, non-preferred answer set, I_4, would vary in the assignments $I_4(cinema) = 0$ and $I_4(beach) = c[2]$. Notice that I_1 and I_3 are analogous in the sense that they switch their preferences orders depending on whether we had *hot* or not, respectively. However, the causal values are not completely analogous: while in I_1 the explanation for *cinema* involves two labels, $h \cdot b[1]$, in I_3 the explanation for *beach* only refers to $c[1]$. This is because the meaning of default negation in causal justifications is understood as a default precondition, rather than an actual, effective cause. To generate a symmetric with respect to I_1 we should encode rule b as:

$$b : beach \times cinema \leftarrow \neg hot$$

and, if fact $\neg hot$ were labelled to trace its effects, say using:

$$g : \neg hot$$

then $I_3(beach)$ would become $g \cdot c[1]$.

It is not difficult to see that program $P_1 \cup \{summer, rain\}$ yields a unique preferred answer set whose causal version, I_5, yields the explanations $I_5(summer) = I_5(rain) = 1$, $I_5(\neg beach) = r$, $I_5(hot) = h$ and $I_5(cinema) = h \cdot c[2]$. Note how, in this case, *cinema* is justified by h (the rule concluding *hot*) and, after that, $c[2]$ meaning that we were forced to select the second choice of rule c (since *beach* was not possible[3]).

In all the previous variations of the example, we never had alternative proofs for an atom: all true atoms have had a unique possible derivation in a given answer set. This is reflected in the fact that we did not get any addition $+$. To illustrate this effect, suppose that whenever we decide making $windsurf$ we always go to the beach whereas, normally, when the day is windy we are in the mood for making surf (if nothing prevents us from doing so). To capture this refinement, assume that P_2 is program P_1 plus rules:

$$\begin{aligned} s: &\quad beach \leftarrow windsurf \\ w: &\ windsurf \leftarrow wind, not\ \neg windsurf \end{aligned}$$

and that we are in a windy day in the summer. The program $P_2 \cup \{wind, summer\}$ has one preferred answer set I_6 whose corresponding causal explanations are $I_6(summer) = I_6(wind) = 1, I_6(windsurf) = w, I_6(hot) = h, I_6(beach) = h \cdot b[1] + w \cdot s$. In other words, we get two explanations for *beach*: the previous one saying that, as it is hot h, we used the first choice of rule b; plus a second one saying that, as we want to make windsurf w, we applied rule s.

5 Preferred explanations

In [10] it was shown how the number of alternative causal explanations for an atom in a positive program may grow exponentially in the worst case. The reason for that is related to the application of distributivity, something we need when we want to express the final atom value in disjunctive normal form (sum of alternative causes). So, as we explained in the introduction, this is another case in which we may have many potential "solutions" (the causal explanations) where we may be interested in showing preferences among them.

One straightforward manner to incorporate preferences among terms in the causal values lattice is adding *axioms* involving the order relation among terms. As an elementary expression, if we add an axiom $r \leq r'$ (that is $r + r' = r'$ or equivalently $r * r' = r$) for a pair of rule labels r, r' we will be able to remove some less preferred explanations from causal terms. For instance, think about the last variation of the example, program P_2. We may be interested in preferring explanations that involve a non-preference rule like s or w over explanations that involve a preference like b. In our case, we could add the axioms:

$$b[1] \leq s \qquad b[2] \leq s \qquad b[1] \leq w \qquad b[2] \leq w$$

As a result, we would be able to prove that:

$$b[1] + w \cdot s = (\underbrace{b[1] + w}_{=w}) \cdot (\underbrace{b[1] + s}_{=s}) = w \cdot s$$

[3] Remember that we implicitly assume the existence of constraint $\bot \leftarrow beach, \neg beach$.

that is, $b[1] \leq w \cdot s$. Since $h \cdot b[1] \leq b[1]$ by absorption, we conclude $h \cdot b[1] \leq w \cdot s$ and, thus, $I_6(beach) = h \cdot b[1] + w \cdot s = w \cdot s$ removing, in this way, the less preferred explanation based on rule b.

Many different preference criteria can be thought for selecting the "best" explanations. For instance, it makes sense preferring an explanation that uses the i-th choice of an ordered disjunction than one that uses a j-th choice with $i < j$. To capture that behaviour, we could define the degree n of a causal term without sums as the maximum value k occurring in any label of the form $r[k]$ (0 if no label of that form occurs), and add axiom schemata to force that causal explanations with higher degree are smaller under the causal order relation. As an example, consider the following LPOD:

$$
\begin{aligned}
a &: p \times q \times r \\
b &: t \times h \times p \times q \\
c &: h \times t \times q \\
\bot &\leftarrow p \\
\bot &\leftarrow h \\
\bot &\leftarrow t
\end{aligned}
$$

The preferred answer set makes q true with the explanation $I(q) = a[2] + b[4] + c[3]$ that, under the criterion mentioned above, would collapse to the explanation with smallest degree $I(q) = a[2]$. In other words, the "strongest" reason for q is that we took the second choice of rule a.

6 Conclusions and open topics

In this paper we have presented a first exploratory attempt to provide explanations or justifications for the preferred answer sets obtained from Logic Programs with Ordered Disjunction (LPODs). At a same time, we have also discussed the possibility of incorporating preferences when there exist alternative explanations for a same atom.

The current approach is a simple first step towards the final goal of providing explanations of preferred answer sets and helping the user to refine her formal representation. There are many open topics that are interesting for future work. As a first example, when explaining the outcome of an LPOD, the approach in this paper only provides the explanation of true atoms in a given preferred answer set. This may help us to find out where did the information contained in one preferred choice come from. However, in many cases, the question of why a given fact or literal, that we know that may be true in another solution to our problem (i.e., some answer set perhaps not preferred) has not been eventually true in some preferred answer set. Answering questions of the form "why-not" has been studied in [11] for a different algebraic approach and the incorporation of this type of queries to causal justifications is currently under study. Formally, this will involve the incorporation of a negation operator in the algebra of causal values. Still, even if we are eventually able of answering "why-not" questions, the case of LPODs introduces an additional difficulty since what we would probably want is to

know how the set of preferences has prevented that a literal (possible in another answer set) became true in any preferred solution.

Regarding the topic on preferring explanations, the addition of a negation operator to the causal algebra could also be interesting to express preferences like, for instance, saying that any positive proof is preferred over a proof that depends on negation (a default). In the windsurf variation of our running example, this would mean that adding a rule of the form:

$$m : beach \leftarrow romantic$$

together with the fact $t : romantic$ should provide a stronger explanation than $w \cdot s$ since $t \cdot m$ has not applied any default, whereas w was actually a default rule.

References

1. Gerhard Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989*, pages 1043–1048, 1989.
2. Gerhard Brewka. Reasoning about priorities in default logic. In *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 2.*, pages 940–945, 1994.
3. Gerhard Brewka. 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.*, pages 100–105, 2002.
4. Gerhard Brewka and Thomas Eiter. Preferred answer sets for extended logic programs. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy, June 2-5, 1998.*, pages 86–97, 1998.
5. Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.
6. Gerhard Brewka, Ilkka Niemelä, and Tommi Syrjänen. Logic programs with ordered disjunction. *Computational Intelligence*, 20(2):335–357, 2004.
7. Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski. Preferences and nonmonotonic reasoning. *AI Magazine*, 29(4):69–78, 2008.
8. Pedro Cabalar. A logical characterisation of ordered disjunction. *AI Communications*, 24(2):165–175, 2011.
9. Pedro Cabalar, Jorge Fandinno, and Michael Fink. Causal graph justifications of logic programs. *Theory and Practice of Logic Programming*, 14(4–5):603–618, 2014. special issue on ICLP'14.
10. Pedro Cabalar, Jorge Fandinno, and Michael Fink. A complexity assessment for queries involving sufficient and necessary causes. In *Proc. of the 14th European Conference on Logics in Artificial Intelligence (JELIA'14)*, volume 8761 of *Lecture Notes in Artificial Intelligence*, pages 300–310. Springer, 2014.
11. C. V. Damásio, A. Analyti, and G. Antoniou. Justifications for logic programming. In *Proc. of the 12th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning, (LPNMR'13)*, volume 8148 of *Lecture Notes in Computer Science*, pages 530–542. Springer, 2013.
12. Marc Denecker and Danny De Schreye. Justification semantics: A unifiying framework for the semantics of logic programs. In *Proc. of the Logic Programming and Nonmonotonic Reasoning Workshop*, pages 365–379, 1993.

13. M. Gebser, J. Pührer, T. Schaub, and H. Tompits. Meta-programming technique for debugging answer-set programs. In *Proc. of the 23rd Conf. on Artificial Inteligence (AAAI'08)*, pages 448–453, 2008.
14. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. A. Bowen, editors, *Logic Programming: Proc. of the Fifth International Conference and Symposium (Volume 2)*, pages 1070–1080. MIT Press, Cambridge, MA, 1988.
15. Arend Heyting. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse*, pages 42–56, 1930.
16. V. Marek and M. Truszczyński. *Stable models and an alternative logic programming paradigm*, pages 169–181. Springer-Verlag, 1999.
17. I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25:241–273, 1999.
18. David Pearce. A new logical characterisation of stable models and answer sets. In *Non monotonic extensions of logic programming. Proc. NMELP'96. (LNAI 1216)*. Springer-Verlag, 1996.
19. Luís Moniz Pereira, Joaquim Nunes Aparício, and José Júlio Alferes. Derivation procedures for extended stable models. In John Mylopoulos and Raymond Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 863–869. Morgan Kaufmann, 1991.
20. E. Pontelli, T. C. Son, and O. El-Khatib. Justifications for logic programs under answer set semantics. *Theory and Practice of Logic Programming*, 9(1):1–56, 2009.
21. Raymond Reiter. A logic for default reasoning. *Artif. Intell.*, 13(1-2):81–132, 1980.
22. C. Schulz, M. Sergot, and F. Toni. Argumentation-based answer set justification. In *Proc. of the 11th Intl. Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense'13)*, 2013.
23. Maarten H. van Emden and Robert A. Kowalski. The semantics of predicate logic as a programming language. *J. ACM*, 23(4):733–742, 1976.
24. Joost Vennekens. Actual causation in cp-logic. *TPLP*, 11(4-5):647–662, 2011.