*TLP* **16** (5–6): 587–603, 2016. © Cambridge University Press 2016 
doi:10.1017/S1471068416000454

# *Justifications for programs with disjunctive and causal-choice rules**

PEDRO CABALAR and JORGE FANDINNO

*Department of Computer Science*
*University of Corunna, Corunna, Spain*
(*e-mail:* {cabalar, jorge.fandino}@udc.es)

*submitted 6 May 2016; revised 8 July 2016; accepted 22 August 2016*

**Abstract**

In this paper, we study an extension of the stable model semantics for disjunctive logic programs where each true atom in a model is associated with an algebraic expression (in terms of rule labels) that represents its justifications. As in our previous work for non-disjunctive programs, these justifications are obtained in a purely semantic way, by algebraic operations (product, addition and application) on a lattice of causal values. Our new definition extends the concept of *causal stable model* to disjunctive logic programs and satisfies that each (standard) stable model corresponds to a disjoint class of causal stable models sharing the same truth assignments, but possibly varying the obtained explanations. We provide a pair of illustrative examples showing the behaviour of the new semantics and discuss the need of introducing a new type of rule, which we call *causal-choice*. This type of rule intuitively captures the idea of "*A* may cause *B*" and, when causal information is disregarded, amounts to a usual choice rule under the standard stable model semantics.

## 1 Introduction

The strong connection between Non-Monotonic Reasoning (NMR) and Logic Programming (LP) semantics for default negation has made possible that LP tools became nowadays an important paradigm for Knowledge Representation (KR) and problem-solving in Artificial Intelligence (AI). In particular, *Answer Set Programming* (ASP) (Niemelä 1999; Marek and Truszczyński 1999) is an LP paradigm based on the *stable models semantics* (Gelfond and Lifschitz 1988; Gelfond and Lifschitz 1991) that has become established as a preeminent framework for practical NMR with applications in diverse areas of AI including planning, reasoning about actions, diagnosis, abduction and beyond (Baral 2003). An important difference between most LP semantics (including the stable model semantics) and classical models (or even some other NMR approaches) is that true atoms in LP must be founded or justified by a given derivation. These *justifications* are not provided in the semantics itself, but can be syntactically built in some way in terms of the program rules, as

* This research was partially supported by Spanish Project TIN2013-42149-P.

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

studied in several approaches (Specht 1993; Pemmasani et al. 2004; Pontelli et al. 2009; Denecker et al. 2015; Schulz and Toni 2016).

Rather than manipulating justifications as mere syntactic objects, two recent approaches have considered extended multi-valued semantics LP extensions where justifications are treated as *algebraic* constructions in terms of rule labels: *Why-not Provenance* (Damásio et al. 2013) and *Causal Graph Justifications* (CJ) (Cabalar et al. 2014a). For instance, as an illustration of the latter, consider the following positive program $P_1$

$$r_1 : \quad dead \leftarrow shoot \tag{1}$$

$$r_2 : \quad shoot \leftarrow harvey \tag{2}$$

$$\quad harvey \tag{3}$$

It has been showed in (Cabalar et al. 2014a) that program $P_1$ has a least causal model $I$, which not only makes *dead* true, but also assigns the expression $I(dead) = harvey \cdot r_2 \cdot r_1$ capturing the successive application of $r_2$ and then $r_1$. The availability of such information on justifications *inside the models* of a program can be of great interest for Knowledge Representation (KR) since it allows defining new types of literals to inspect the causal relations among derived atoms.

For instance, (Fandinno 2015b) introduced constructs like:

$$prison(harvey) \leftarrow harvey \quad \texttt{sufficient for} \quad dead \tag{4}$$

where '`sufficient for`' decides whether *harvey* has been a sufficient cause for *dead* by completely relying on the semantic definitions, rather than making a syntactic analysis of the program. Note that avoiding the latter is crucial to achieve an elaboration tolerant representation, since causation may be affected by multiple levels of indirect effects or the interplay with defaults like inertia. (Fandinno 2016) extends this formalism allowing the use of '`necessary for`' and '`contributed to`' to capture whether *harvey* has been necessary for (resp. contributed to) *dead*. To the best of our knowledge, no other formalism integrates this kind of literals in LP or allows deriving new conclusions from information about cause-effect relations. Unfortunately, CJ is only defined for non-disjunctive logic programs – this holds, in fact, for all the approaches aforementioned. This is obviously a drawback from an ASP point of view, since no CJ justifications were defined for programs with disjunction, but also, from a KR point of view, since disjunction is a natural way for expressing non-deterministic causation.

In this paper, we investigate an extension of the CJ semantics for disjunctive logic programs, in which, per each standard stable model, we will obtain a class of causal stable models providing explanations for the derived (true) atoms. As an example, let program $P_2$ be the result of replacing (2) in $P_1$ by the following rules

$$r_2 : \quad shoot \leftarrow tails \tag{5}$$

$$r_3 : \quad head \vee tails \leftarrow harvey \tag{6}$$

stating that Harvey throws a coin and only shoots when he gets tails. This program has two standard stable models: $\{harvey, head\}$ and $\{harvey, tails, shoot, dead\}$. In

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

the latter, the explanation for *dead* will have the form

$$harvey \cdot r_3^{tails} \cdot r_2 \cdot r_1 \tag{7}$$

where $r_3^{tails}$ points out that the disjunct *tails* in $r_3$ has been effectively applied.

We summarise our contributions as follows.

- We define a multi-valued semantics for disjunctive logic programs[1] in which each atom $A$ is associated to an algebraic expression formed by rule labels and three operations: a product '$*$' representing conjunction or joint causation; an addition '$+$' separating alternative causes; and a non-commutative product '$\cdot$' that stands for rule application. We show that there is a one-to-one correspondence between the standard stable models and a corresponding class of *causal stable models* that provide explanations for each true atom in each model (Section 3).
- We show that disjunctive rules are not enough to capture informal statements like "$A$ may cause $B$" and introduce the notion of *causal-choice* rule to represent this kind of statement. When causal information is disregarded, causal-choice rules amount to usual choice rules under the standard stable model semantics (Section 4).

The rest of the paper is organised as follows. The next section revises the standard stable model semantics for disjunctive logic programs and recalls the causal algebra introduced in (Cabalar et al. 2014a). Section 3 provides the new definition of causal stable models for disjunctive programs and shows some correspondence properties with standard stable models. In Section 4, we introduce causal-choice rules, explain their behaviour with an example and, again, provide a correspondence with standard ASP. Section 5 revises some related approaches and, finally, Section 6 concludes the paper. Proofs of formal results from the paper can be found in an extended version (Cabalar and Fandinno 2016b).

## 2 Preliminaries

We start by recalling some standard LP definitions, with the only extension of rule labels. A *signature* is a pair $\langle At, Lb \rangle$ of sets that respectively represent a set of *atoms* (or *propositions*) and a set of *labels*. As usual, a *literal* is defined as an atom $A$ (positive literal) or its default negation $not\ A$ (negative literal).

*Definition 1 (Program)*
Given a signature $\langle At, Lb \rangle$, a *rule* is an expression of the form

$$r_i : \quad A_1 \vee \ldots \vee A_l \vee not\ A_{l+1} \vee \ldots \vee not\ A_k \quad \leftarrow \quad B_1, \ldots, B_m, not\ B_{m+1}, \ldots, not\ B_n \tag{8}$$

where $r_i \in Lb$ is a label and all $A_j \in At$ and $B_j$ are atoms with $0 \leqslant l \leqslant k$ and $0 \leqslant m \leqslant n$. A *program* $P$ is a set of rules.

[1] For simplicity, in this paper, we will not consider causal literals as in rule (4), but the extension can be done in a similar way as (Fandinno 2015b) extended (Cabalar et al. 2014a).

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

Given a rule $r$ of the form of (8), by $head^+(r) \stackrel{\text{def}}{=} \{A_1 \dots A_l\}$ and $head^-(r) \stackrel{\text{def}}{=} \{A_{l+1} \dots A_k\}$, we respectively denote the positive and negative head of $r$. By $head(r) \stackrel{\text{def}}{=} head^+(r) \cup not\, head^-(r)$, we denote the whole head of $r$ where $not\, head^-(r)$ stands for the set $\{\ not\, A \mid A \in head^+(r)\ \}$. By $body^+(r) \stackrel{\text{def}}{=} \{B_1, \dots, B_m\}$ and $body^-(r) \stackrel{\text{def}}{=} \{B_{m+1}, \dots, B_n\}$, we respectively denote the positive and negative body of $r$ and, by $body(r) \stackrel{\text{def}}{=} body^+(r) \cup not\, body^-(r)$, its whole body. A rule $r$ is said to be *normal* iff its head is a single positive literal, that is, $head(r) = head^+(r) = \{A\}$ and it is said to be *positive* iff all literals in it are positive, that is, $head^-(r) = body^-(r) = \emptyset$. A normal rule with empty body, $body(r) = \emptyset$, is called a *fact* and we usually represent it omitting the body and sometimes the symbol '$\leftarrow$.' Furthermore, we will also allow atom names to be used as labels, that is, $At \subseteq Lb$, and that fact $A$ in a program actually stands for the labelled rule $(A : A \leftarrow)$. Any rule that is not a fact has a label from $Lb_{\not\!A} \stackrel{\text{def}}{=} Lb \backslash At$. When $head(r) = \emptyset$, we say that the rule $r$ is a *constraint* and we may omit the rule label. A program $P$ is said to be *normal* or *positive* when all its rules are normal or positive, respectively.

We say that a set of atoms $S \subseteq At$ is *closed* under $P$ if and only if, for every rule $r \in P$, $head^+(r) \cap S \neq \emptyset$ whenever $body^+(r) \cup head^-(r) \subseteq S$ and $body^-(r) \cap S = \emptyset$. We recall next the definitions of reduct and stable model. The *reduct* of a program $P$ w.r.t. a set of atoms $S \subseteq At$, in symbols $P^S$, is the result of

1. removing all rules such that $B \in S$ for some $B \in body^-(r)$,
2. removing all rules such that $B \notin S$ for some $B \in head^-(r)$,
3. removing all the negative literals for the remaining rules.

A set of atoms $S \subseteq At$ is a *GL-stable model* of a program $P$ iff it is a $\subseteq$-minimal closed set under the positive program $P^S$.

We will also need the following definitions from (Cabalar et al. 2014a).

*Definition 2 (Term)*
Given a set of labels $Lb$, a *term* $t$ is recursively defined as one of the following expressions

$$t \quad ::= \quad l \ \Big|\ \prod S \ \Big|\ \sum S \ \Big|\ t_1 \cdot t_2$$

where $l \in Lb$ is a label, $t_1, t_2$ are in their turn terms and $S$ is a (possibly empty and possibly infinite) set of terms.

When $S = \{t_1, \dots, t_n\}$ is a finite set, we will write $t_1 * \dots * t_n$ and $t_1 + \dots + t_n$ instead of $\prod S$ and $\sum S$, respectively. The empty sum and empty product are respectively represented as 0 and 1. We assume that application '$\cdot$' has higher priority than product '$*$' and, in its turn, product '$*$' has higher priority than addition '$+$'. As commented in the introduction, product '$*$' represents join causation. Consider for instance a program $P_3$ obtained by adding the fact *loaded* to program $P_1$ and replacing rule (1) by

$$r_1 : dead \leftarrow shoot, loaded \tag{9}$$

Program $P_3$ will assign the expression $(loaded * harvey \cdot r_2) \cdot r_1$ to the atom *dead*. On the other hand, addition '$+$' is used to capture alternative causes. As an illustration,

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

| *Associativity* | *Absorption* | *Identity* | *Annihilator* |
|---|---|---|---|
| $t \cdot (u \cdot w) = (t \cdot u) \cdot w$ | $t = t + u \cdot t \cdot w$<br>$u \cdot t \cdot w = t * u \cdot t \cdot w$ | $t = 1 \cdot t$<br>$t = t \cdot 1$ | $0 = t \cdot 0$<br>$0 = 0 \cdot t$ |

| *Idempotence* | *Addition distributivity* | *Product distributivity* |
|---|---|---|
| $l \cdot l = l$ | $t \cdot (u+w) = (t \cdot u) + (t \cdot w)$<br>$(t+u) \cdot w = (t \cdot w) + (u \cdot w)$ | $c \cdot d \cdot e = (c \cdot d) * (d \cdot e)$ with $d \neq 1$<br>$c \cdot (d * e) = (c \cdot d) * (c \cdot e)$<br>$(c * d) \cdot e = (c \cdot e) * (d \cdot e)$ |

Fig. 1. Properties of the '·'operators: $t, u, w$ are terms, $l$ is a label and $c, d, e$ are terms without addition '+'. Addition and product distributivity are also satisfied over infinite sums and products.

if we add the rules:

$$joker_stab \tag{10}$$

$$r_4 : dead \leftarrow joker_stab \tag{11}$$

to program $P_1$, meaning that the Joker stabs the victim, apart from the simultaneous Harvey's shot, then this new program $P_4$ will assign the expression $(loaded * harvey \cdot r_2) \cdot r_1 + joker_stab \cdot r_4$ reflecting the existence of two alternative and independent causes.

*Definition 3 (Value)*
*(Causal) values* are the equivalence classes of terms under axioms for a completely distributive (complete) lattice with meet '$*$' and join '+' plus the axioms of Figure 1. The set of values is denoted by $\mathbf{V}_{Lb}$. Furthermore, by $\mathbf{C}_{Lb}$ we denote the subset of causal values with some representative term without addition '+'.

All three operations, '$*$', '+' and '·' are associative. Product '$*$' and addition '+' are also commutative, and they satisfy the usual absorption and distributive laws with respect to infinite sums and products of a completely distributive lattice. The lattice order relation is defined as:

$$t \leqslant u \qquad \text{iff} \qquad t * u = t \qquad \text{iff} \qquad t + u = u$$

An immediate consequence of this definition is that the $\leqslant$-relation has the product as g.l.b., the addition as l.u.b., 1 as top element and 0 as bottom element. Term 1 represents a value which holds by default, without an explicit cause, and will be assigned to the empty body. Term 0 represents the absence of cause or the empty set of causes, and will be assigned to falsity. Furthermore, applying distributivity (and absorption) of products and applications over addition, every term can be represented in a *(minimal) disjunctive normal form* in which addition is not in the scope of any other operation and every pair of addends are pairwise $\leqslant$-incomparable. This normal form emphasizes the intuition that addition '+' separates alternative causes. Consider, for instance, a program $P_5$ containing the rule

$$r_4 : no_heartbeat \leftarrow dead \tag{12}$$

plus program $P_4 = \{(1), (2), (3), (10), (11)\}$. Program $P_5$ has a least model that coincides with $I_4$ of $P_4$ in all atoms excepting for the new atom *no_heartbeat*, whose justification

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

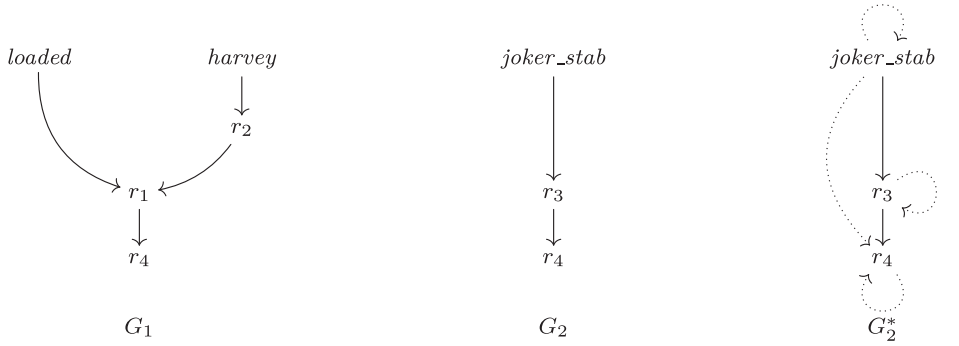



Fig. 2. Graphs $G_1$ and $G_2$ representing the two alternative causes of *no_heartbeat* in program $P_5$. Graph $G_2^*$ is the reflexive and transitive closure of $G_2$ and, thus, the c-graph associated to $G_2$.

becomes:

$$I_5(no_heartbeat) = I_5(dead)\cdot r_4 \quad (13)$$

$$= \big(\,(loaded * harvey\cdot r_2)\cdot r_1 \;+ joker_stab\cdot r_4\cdot r_3\,\big)\cdot r_4 \quad (14)$$

$$= (loaded * harvey\cdot r_2)\cdot r_1\cdot r_4 + \;\; joker_stab\cdot r_3\cdot r_4 \quad (15)$$

Expression (15) is in minimal disjunctive normal and emphasizes the existence of two alternative causes. In this sense, any term without addition $G \in \mathbf{C}_{Lb}$ represents an individual cause, while terms with addition $t \in \mathbf{V}_{Lb}\backslash\mathbf{C}_{Lb}$ will be used to represent sets of non-redundant causes.

Furthermore, each individual cause $G \in \mathbf{C}_{Lb}$ can be depicted as a graph whose vertices are labels. For instance, graphs $G_1$ and $G_2$ in Figure 2 correspond to the two individual causes of the atom *no_heartbeat* with respect to the least model of $P_5$. This correspondence was formalised in (Fandinno 2015a) in the following way.

*Definition 4 (Causal graph)*
Given a set of labels *Lb*, a *causal graph (or c-graph)* $G \subseteq Lb \times Lb$ is a set of edges transitively and reflexively closed. By $\mathbf{G}_{Lb}$ we denote the set of all c-graphs that can be formed with labels from *Lb*. Furthermore, given two causal graphs $\{G, G'\} \subseteq \mathbf{G}_{Lb}$, by $G \cdot G'$ and $G * G'$, we denote the transitive closure of $G \cup G' \cup \{\ (v,v') \mid (v,v) \in G \text{ and } (v',v') \in G'\ \}$ and $G \cup G'$, respectively.

*Theorem 1 (From Fandinno 2015a)*
The function $term : \mathbf{G}_{Lb} \longrightarrow \mathbf{C}_{Lb}$ given by

$$term(G) \;\mapsto \prod_{(v_1,v_2)\in G} v_1\cdot v_2$$

is an isomorphism between algebras $\langle \mathbf{G}_{Lb}, *, \cdot, \emptyset, \subseteq\rangle$ and $\langle \mathbf{C}_{Lb}, *, \cdot, 1, \leqslant\rangle$.

Defining causal graphs as transitively and reflexively closed is convenient for a simpler definition of product and application over graphs. However, for the sake of readability, we will just depict a causal graph $G$ as one of its transitive and

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

reflexive reductions[2]. For instance, Figure 2 shows a graph $G_2$ that will actually stand for its closure, the causal graph $G_2^*$. Furthermore, we will also allow writing $I_5(no_heartbeat) = G_1 + G_2$ instead of (15). It is also worth to mention that each path in a causal graph may be understood as causal chain in the sense of Lewis (1973): a finite sequence of actual events $e_1, \ldots, e_n$ where each $e_i$ causally depends on $e_{i-1}$ (for $0 < i \leqslant n$).

## 3 Causal stable models for disjunctive logic programs

An *interpretation* is a mapping $I : At \longrightarrow \mathbf{V}_{Lb}$ assigning a value to each atom. An interpretation $I$ is said to be *standard* or *two-valued* when it maps each atom into the set $\{0, 1\}$. The value assigned to a negative literal $not\, A$ by an interpretation $I$, denoted as $I(not\, A)$, is defined as: $I(not\, A) \stackrel{\text{def}}{=} 1$ if $I(A) = 0$; and $I(not\, A) \stackrel{\text{def}}{=} 0$ otherwise. For any pair of interpretations $I$ and $J$, we write $I \leqslant J$ to represent the straightforward causal ordering, that is, $I(A) \leqslant J(A)$ for every atom $A \in At$.

Since body disjunction can be captured by addition '+' (Fandinno 2015b), an immediate attempt for interpreting disjunctive programs would consist in just defining $I(A \vee A') = I(A) + I(A')$ and selecting $\leqslant$-minimal models. However, this naive approach does not capture the intending meaning. Consider for instance the following program $P_6$

$$r_1 : \quad head \vee tails \leftarrow toss \qquad\qquad a : \quad toss \qquad\qquad b : \quad toss$$

where two persons $a$ and $b$ both give the command to *toss* a coin. One may expect that program $P_6$ has two causal stable models, one $I_6$ in which $I_6(head) = a{\cdot}r_1^{head} + b{\cdot}r_1^{head}$ and $I_6(tails) = 0$, and another $I_6'$ in which $I_6'(head) = 0$ and $I_6'(tails) = a{\cdot}r_1^{tails} + b{\cdot}r_1^{tails}$. However, under this naive definition, there are other two additional causal stable models that unintendedly "distribute" the causes of *toss* among *head* and *tails*. We get, for instance, $I_6''$ where $I_6''(head) = a{\cdot}r_1^{head}$ and $I_6''(tails) = b{\cdot}r_1^{tails}$, and a dual model $I_6'''$ that switches the roles of $a$ and $b$.

This example shows that, rather than taking the sum of all disjuncts in the head, the evaluation of rules should consider each head disjunct independently, as defined below.

*Definition 5 (Model)*
An *interpretation* $I$ satisfies rule $r$ of the form of (8) iff

$$\big(I(B_1) * \ldots * I(B_m) * I(not\, B_{m+1}) * \ldots * I(not\, B_n)\big) \cdot r_i \cdot A_j \quad \leqslant \quad I(A_j) \qquad (16)$$

for some atom $A_j \in head(r)$. We say that an interpretation $I$ is a *model* of a program $P$, in symbols $I \models P$, iff $I$ satisfies all rules in $P$. Moreover, if $I$ is standard (two-valued) we further write $I \models^{st} P$ and call it *standard model.*

[2] Recall that the transitive and reflexive reduction of a graph $G$ is a graph $G'$ whose transitive and reflexive closure is $G$. A causal graph, in which every cycle is a reflexive edge, has a unique transitive and reflexive reduction.

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

*Observation 1*
*If $r$ is a fact $A$, that is, it has the form $(A : A \leftarrow)$ then $I \models r$ iff $I(A) \geqslant A \cdot A = A$ (by idempotence of application on labels).*

It is easy to see that, under Definition 5, interpretations $I_6''$ and $I_6'''$ are no longer models of $P_6$. Still, a second issue comes from selecting $\leqslant$-minimal models. Consider, for instance, the following program $P_7$

$$r_1 : \quad head \vee tails \qquad\qquad head$$

which has two $\leqslant$-minimal models, one in which $I_7(head) = head + r_1^{head}$ and $I_7(tails) = 0$, plus another in which $I_7'(head) = head$ and $I_7'(tails) = r_1^{tails}$. However, only the former corresponds to the unique GL-stable model of this program, which leaves *tails* false under its truth-minimality criterion. To overcome this issue, we will define an extra partial order $\sqsubseteq$. First, for any interpretation $I$ we define its corresponding standard interpretation, $I^{st}$, as follows:

$$I^{st}(A) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{iff } I(A) > 0 \\ 0 & \text{iff } I(A) = 0 \end{cases}$$

We define the set of true atoms in an interpretation $I$ as $Atoms(I) \stackrel{\text{def}}{=} \{\, A \in At \mid I(A) \neq 0 \,\}$.

*Proposition 1*
*For any standard interpretation $I$ and positive program $P$: $I \models^{st} P$ iff $Atoms(I)$ is closed under $P$.*

Then, for any pair of interpretations $I$ and $J$, we write $I \sqsubseteq J$ when either $I \leqslant J$ or $Atoms(I) \subset Atoms(J)$. That is, $I \sqsubseteq J$ is a weaker relation, since apart from the cases in which $I \leqslant J$ it also holds when true atoms in $I$ are a strict subset of true atoms in $J$. As usual, we write $I < J$ (resp. $I \sqsubset J$) iff $I \leqslant J$ (resp $I \sqsubseteq J$) and $I \neq J$. Notice that $Atoms(I) \subset Atoms(J)$ implies $I \neq J$ and so $I \sqsubset J$. We say that an interpretation $I$ is $\leqslant$-minimal (resp. $\sqsubseteq$-minimal) satisfying some property when there is no $J < I$ (resp. $J \sqsubset I$) satisfying that property. It is worth to notice that there is a $\leqslant$-bottom and $\sqsubseteq$-bottom interpretation $\mathbf{0}$ (resp. a $\leqslant$-top and $\sqsubseteq$-top interpretation $\mathbf{1}$) that stands for the interpretation mapping every atom $A$ to 0 (resp. 1).

*Definition 6 (Causal/standard stable model of a positive program)*
Let $P$ be a positive program. A model $I$ of $P$ is a *causal stable* model iff it is $\sqsubseteq$-minimal among models of $P$. A standard model $I$ of $P$ is a *standard stable* model iff $I$ is $\sqsubseteq$-minimal among standard models of $P$.

Standard stable models have a straightforward relation to GL-stable models, as stated below:

*Theorem 2*
Let $P$ be a positive program. An interpretation $I$ is a standard stable model of $P$ iff $Atoms(I)$ is a GL-stable model of $P$.

From now on we use both concepts indistinctly. Note that not any standard stable model $I$ of $P$ needs to be causal stable, since there could exist another non-standard

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

model $J \sqsubset I$. Still, there exists a strong connection between standard and causal stable models.

*Theorem 3*
Let $P$ be a positive program. Then, a standard interpretation $J$ is a standard stable model of $P$ iff there is some causal stable model $I$ of $P$ such that $I^{st} = J$.

In other words, there is a one-to-one correspondence between each standard stable model and a class of causal stable models that agree in the truth assignment of atoms (but may vary in their explanations). For instance, $P_2$=$\{(1),(3),(5),(6)\}$ has two GL-stable models, $S_2 = \{harvey, head\}$ and $S_2' = \{harvey, tails, shoot, dead\}$. Notice that any model $I_2$ of program $P_2$ must satisfy $I_2(harvey) \geqslant harvey$ and, thus, any $\sqsubseteq$-minimal model must satisfy that $I_2(harvey) = harvey$. Similarly, any model $I_2$ must satisfy either $I_2(head) \geqslant harvey{\cdot}r_1{\cdot}head$ or $I_2(tails) \geqslant harvey{\cdot}r_1{\cdot}tails$ and, thus, there are two $\sqsubseteq$-minimal models, $I_2$ and $I_2'$, of $P_2$ satisfying:

$$\begin{array}{lcllcl} I_2(harvey) & = & harvey & I_2'(harvey) & = & harvey \\ I_2(head) & = & harvey{\cdot}r_1{\cdot}head & I_2'(head) & = & 0 \\ I_2(tails) & = & 0 & I_2'(tails) & = & harvey{\cdot}r_1{\cdot}tails \\ I_2(shoot) & = & 0 & I_2(shoot) & = & harvey{\cdot}r_1{\cdot}tails{\cdot}r_2{\cdot}shoot \\ I_2(dead) & = & 0 & I_2(dead) & = & harvey{\cdot}r_1{\cdot}tails{\cdot}r_2{\cdot}shoot{\cdot}r_3{\cdot}dead \end{array}$$

For the sake of clarity, we will write $r_i^A$ instead of $r_i{\cdot}A$ when $r_i \in Lb_{\mathscr{R}}$ and $A \in At$. Intuitively, symbol $r_i^A$ means that among the possible consequences of rule $r_i$, it has caused $A$. For normal rules like $r_1$ and $r_2$ stating the selected atom is redundant and, thus, we will usually omit the head atom label. Using these conventions, we may rewrite the $\sqsubseteq$-minimal models of $P_2$ as

$$\begin{array}{lcllcl} I_2(harvey) & = & harvey & I_2'(harvey) & = & harvey \\ I_2(head) & = & harvey{\cdot}r_1^{head} & I_2'(head) & = & 0 \\ I_2(tails) & = & 0 & I_2'(tails) & = & harvey{\cdot}r_1^{tails} \\ I_2(shoot) & = & 0 & I_2'(shoot) & = & harvey{\cdot}r_1^{tails}{\cdot}r_2 \\ I_2(dead) & = & 0 & I_2'(dead) & = & harvey{\cdot}r_1^{tails}{\cdot}r_2{\cdot}r_3 \end{array}$$

Note that $I_2'(dead)$ is just the justification of *dead* mentioned in the introduction (7). Another interesting observation is that the sets of true atoms in each model, $Atoms(I_2) = \{harvey, head\}$ and $Atoms(I_2') = \{harvey, tails, shoot, dead\}$, respectively corresponding to the two GL-stable models of $P_2$. This correspondence, however, is not always one-to-one, as stated by Theorem 3. To see why, consider a program $P_8$ consisting of the following rules

$$r_1: \quad a \vee b \leftarrow \qquad\qquad r_2: \quad a \leftarrow b \qquad\qquad r_3: \quad b \leftarrow a$$

Program $P_8$ has a unique GL-stable model $\{a, b\}$ but two causal stable models $I_8$ and $I_8'$ such that:

$$\begin{array}{lcllcl} I_8(a) & = & r_1^a & I_8'(a) & = & r_1^b{\cdot}r_2 \\ I_8(b) & = & r_1^a{\cdot}r_3 & I_8'(b) & = & r_1^b \end{array}$$

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

As we can see, the true atoms in both models $Atoms(I) = Atoms(I') = \{a, b\}$ coincide with the unique GL-stable model, but their *explanations differ*. In $I_8$, atom $a$ is a (non-deterministic) effect of the disjunction $r_1$, while $b$ is derived from $a$ through $r_3$. Analogously, $I'_8$ makes $b$ true because of $r_1$ and then obtains $a$ from $b$ through $r_2$.

***Programs with negation.*** To introduce default negation, let us consider a variation of our running example in which shooting the victim may fail in several ways: the victim may be wearing a *bulletproof* vest, the gunpowder may be *wet*, etc. A possible encoding of this scenario is program $P_9$ obtained from program $P_2$=$\{(1), (3), (5), (6)\}$ after replacing rule (1) by

$$r_1 : dead \leftarrow shoot, not\ ab \tag{17}$$

and rules for every possible exception in the following way

$$r_5 : ab \leftarrow wet \tag{18}$$

$$r_6 : ab \leftarrow bulletproof \tag{19}$$

From a causal perspective, saying that the lack of an exception is part of a cause (e.g., for *dead*) may seem rather counterintuitive. It is not the case that we are *dead* because the gunpowder was not *wet*, or because we are not wearing a *bulletproof* vest, or whatever other possible exception that might be added in the future[3]. Instead, as nothing violated default (17), the justifications for *dead* should be the same as in program $P_2$. In this sense, falsity in understood as the *default situation* that is broken when a cause is found[4]. This interpretation carries over to negative literals, so that the presence of *not A* in a rule body does not propagate causal information, but instead, it simply checks the absence of an exception. To capture this behaviour, we proceed to extend the traditional program reduct (Gelfond and Lifschitz 1988) to causal logic programs.

*Definition 7 (Program reduct)*
The *reduct* of a program $P$ with respect to an interpretation $I$, in symbols $P^I$, is the result of

1. removing all rules such that $I(B) \neq 0$ for some $B \in body^-(r)$,
2. removing all rules such that $I(B) = 0$ for some $B \in head^-(r)$,
3. removing all the negative literals for the remaining rules.

*Definition 8 (Causal stable model)*
We say that an interpretation $I$ is a *causal stable model* of some program $P$ iff $I$ is a causal stable model of the positive program $P^I$.

*Observation 2*
*Note that $P^I$ coincides with $P^{I^{st}}$, i.e., the reduct does not vary if we just use the standard interpretation $I^{st}$. Moreover, it also coincides with the classical GL-reduct $P^{Atoms(I)}$.*

[3] A case of the well-known *qualification problem* (McCarthy 1987), i.e., the impossibility of listing all the possible conditions that prevent an action to cause a given effect.

[4] The paper (Hitchcock and Knobe 2009) contains an extended discussion with several examples showing how people ordinarily understand causes as deviations from a norm.

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

*Theorem 4*
Let $P$ be a labelled program. Then, $J$ is a standard stable model (i.e. $Atoms(J)$ is a GL-stable model) of $P$ iff there is some causal stable model $I$ of $P$ such that $I^{st} = J$ (i.e. $Atoms(I) = Atoms(J)$).

Theorem 4 is an immediate consequence of Observation 2 plus Theorem 3. Furthermore, it is easy to see that program $P_9$ has the same two standard stable models of $P_2$. The reduct of $P_9$ with respect to interpretations $I_2$ and $I_2'$ is just the result of removing *not ab* from the body of (17). Thus, these are also the two causal stable models of $P_9$. On the other hand, a program $P_{10}$, obtained by adding the fact *wet* to program $P_9$, has two standard stable models $\{harvey, head, wet\}$ and $\{harvey, tails, wet\}$ and two corresponding causal stable models. Note that the reduct of program $P_9$ w.r.t. $I_9$ and $I_9'$ is just the result of removing rule (17) and, thus, there is no justification for *dead* in any of these two causal stable models.

## 4 Causal-choice rules

Disjunctive rules are useful for representing non-deterministic causal laws that represent the possible outcomes of an experiment like throwing a coin. However, disjunction alone is not enough to capture the causal nature of some non-deterministic statements of the form "$A$ may cause $B$." In order to illustrate this limitation, consider the statement "An *infection* may cause the patient to have *fever*." A frequent way to represent this kind of statements in ASP is using a choice rule as follows

$$r_1 : fever \vee not\ fever \leftarrow infection \tag{20}$$

$$infection \tag{21}$$

Program $P_{11}$ consisting of rules (20-21) has two standard stable models $S_{11} = \{fever\}$ and $S_{11}' = \{infection, fever\}$ and, in fact, two respective causal stable models $I_{11}$ and $I_{11}'$:

$$I_{11}(infection) = infection \qquad I_{11}'(infection) = infection$$
$$I_{11}(fever) = 0 \qquad I_{11}'(fever) = infection \cdot r_1^{fever}$$

which obtain the expected result. However, the causal behaviour of (20) is not exactly what we look for but, instead, would correspond to the assertion "*infection* may cause *fever* or *fever* is false." This becomes evident if we add a second external cause for *fever*:

$$r_2 : fever \leftarrow \tag{22}$$

The unique standard stable model of the new program $P_{12}$ formed by $P_{11}$ plus (22) is now $S_{11}'$, since once *fever* is fixed by $r_2$, the choice $r_1$ *is forced to cause fever*. This "forced causation" is irrelevant when we do not consider causal explanations: in fact, under GL-stable models semantics, once we add a fact for *fever*, rule $r_1$ becomes a tautology and can always be safely removed from the program. However, $r_1$ cannot be removed under the causal semantics (since it expresses some relation between

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

*infection* and *fever*) and, as (22) fixes *fever* true for any model $I$, the corresponding reduct of $r_1$ will mandatorily be:

$$r_1 : \mathit{fever} \leftarrow \mathit{infection} \tag{23}$$

Under these circumstances, it is easy to see that we get a *unique* causal stable model $I_{12}$ of program $P_{12}$ that satisfies $I_{12}(\mathit{fever}) = \mathit{infection} \cdot r_1^{\mathit{fever}} + r_2$. In other words, we get now that *infection* is always forced to be one of the causes of *fever*, although the only addition we made was providing an (independent) alternative cause $r_2$.

For capturing the correct meaning of "may cause" in this example, we should have obtained a second causal stable model $I'_{12}$ where $I'_{12}(\mathit{fever}) = r_2$, that is, *fever* is still true due to $r_2$ but the *infection* does not result to be an alternative cause of *fever* this time. The problem with this second model is that while $\mathit{Atoms}(I'_{12}) = \mathit{Atoms}(I_{12})$ means that the reduct for both models will always coincide, and we also have[5] $I'_{12} \sqsubset I_{12}$, meaning that no program may have *both* interpretations as $\sqsubseteq$-minimal models of such a reduct. Hence, it is clear that no program using the previous language can capture the intended models of this example. We introduce next the notion of *causal-choice rule* (and its associated program reduct) that intuitively captures the idea of "$A$ may cause $B$" and that, when causal information is disregarded, amounts to a usual choice rule under the standard stable model semantics.

*Definition 9*
*A* causal-choice rule *is an expression of the form of*

$$r_i : \quad A_1 \vee \ldots \vee A_l \vee \mathrm{not}\, A_{l+1} \vee \ldots \vee \mathrm{not}\, A_k \quad \leftsquigarrow \quad B_1, \ldots, B_m, \mathrm{not}\, B_{m+1}, \ldots, \mathrm{not}\, B_n \tag{24}$$

*where $r_i \in Lb$ is a label and all $A_j \in At$ and $B_j$ are atoms. For every causal-choice rule $r$ of the form of* (24)*, by rule*$(r)$ *we denote its corresponding rule of the form of* (8)*, i.e., rule*$(r)$ *corresponds to replacing '* $\leftsquigarrow$ *' by '* $\leftarrow$ *.'*

*Definition 10*
*The reduct of a program with causal-choice rules $P$, with respect to an interpretation $I$, in symbols also $P^I$, is obtained by*

1. *removing every causal-choice rule $r$ such that $I$ does not satisfy rule*$(r)$*,*
2. *replacing the remaining causal-choice rules $r$ by rule*$(r)$
3. *apply the reduct of Definition 7 to the result of the previous two steps.*

*An interpretation $I$ is a causal stable model of a program with causal-choice rules $P$ iff $I$ is a causal stable model of $P^I$.*

We can now represent the variation of our running example with program $P_{13}$:

$$r_1 : \quad \mathit{fever} \quad \leftsquigarrow \mathit{infection} \tag{25}$$

$$r_2 : \quad \mathit{fever} \quad \leftarrow \tag{26}$$

$$\mathit{infection} \tag{27}$$

[5] Indeed, $I'_{12}(\mathit{fever}) = r_2 < \mathit{infection} \cdot r_1^{\mathit{fever}} + r_2 = I_{12}(\mathit{fever})$ while $I_{12}(\mathit{infection}) = I'_{12}(\mathit{infection}) = \mathit{infection}$.

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

On the one hand, the reduct of program $P_{13}$ w.r.t. $I_{12}$ is the program obtained by replacing the choice rule (25) by its corresponding rule (23). It is easy to check that $I_{12}$ is a $\sqsubseteq$-minimal model of $P_{13}^{I_{12}}$ and, thus, a causal stable model of $P_{13}$. On the other hand, the reduct of program $P_{13}$ w.r.t. $I'_{12}$ is the result of removing rule (25) and, thus, $I'_{12}$ is also a $\sqsubseteq$-minimal model of $P_{13}^{I'_{12}}$ and a causal stable model of program $P_{13}$.

As explained before, this example illustrates that there are programs with causal-choice rules whose causal stable models cannot be captured by programs without them. Despite that, it can be shown that, when causal information is disregarded, causal-choice rules behave like standard choice rules. The following definition and Theorem 5 below formalise this intuitive idea.

*Definition 11*
For any program $P$ and causal-choice rule $r \in P$ of the form of (24), by $\mathtt{ch}(r, P)$, we denote the program obtained after replacing $r$ by a set of rules of the form of

$$r_i : A_1 \vee \ldots \vee A_j \vee not\, A_j \vee \ldots \vee A_l \vee not\, A_{l+1} \vee \ldots \vee not\, A_k \leftarrow body(r) \tag{28}$$

per each atom $A_j \in head^+(r)$.

As an example, the causal-choice (25) in a program $P$ would be replaced by the standard choice (20) in $\mathtt{ch}((25), P)$.

*Theorem 5*
Let $P$ be a program with causal-choice rules, $r \in P$ be a causal-choice rule and let $I$ be a standard interpretation. Then, $I$ is a standard stable model of $P$ iff $I$ is a standard stable model of $\mathtt{ch}(r, P)$.

## 5 Related work

The most obvious related work is our previous approach for non-disjunctive programs (Cabalar et al. 2014a). The following result established a one-to-one correspondence between our causal stable models and the *CJ-stable models* from (Cabalar et al. 2014a).

*Theorem 6*
Let $P$ be a normal program, for every causal stable model $I$, there is a unique CJ-causal stable model $J$ such that $I^{st} = J^{st}$, and vice-versa. Furthermore, if $I$ and $J$ are respectively a causal stable model and CJ-stable model of $P$ such that $I^{st} = J^{st}$, then there is a justification $G' = \lambda(G)$ such that $G' \leqslant J(A)$ for every atom $A$ and justification $G \leqslant I(A)$, where $\lambda(G)$ denotes the result of removing every edge containing an atom label $A \in At$ in the causal graph $G$.

Note, that causal justifications with respect to any causal stable model are always a (possibly strict) subset of the causal justifications with the corresponding CJ-stable model. To illustrate this difference, consider the following normal program $P_{14}$

$$r_1 : a \leftarrow \qquad r_3 : b \leftarrow a \qquad r_4 : c \leftarrow a \qquad r_5 : d \leftarrow b, c$$

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

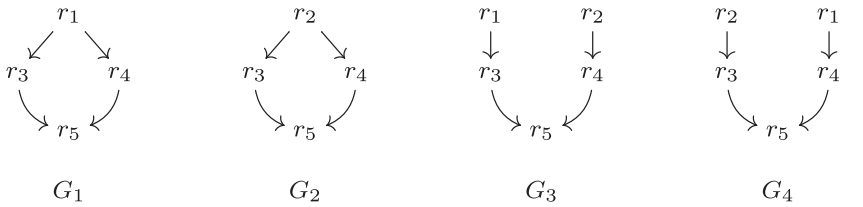



Fig. 3. Causal graphs associated with $d$ by the unique CJ-stable model of program $P_{14}$.

whose unique CJ-stable model satisfies $I_{14}(d) = G_1$ with each $G_1$ being the causal graph $G_1$ depicted in Figure 3. Similarly, the unique causal stable model of program $P_{14}$ satisfies $I_{14}(d) = G_1'$ where $G_1'$ is the result of removing every edge containing an atom label $A \in At$ in the causal graph $G_1$. However, if we consider a program $P_{15}$ obtained by adding rule $(r_2 : a \leftarrow )$ to program $P_{14}$, then this program has a unique CJ-stable model that satisfies $J_{15}(d) = G_1 + G_2 + G_3 + G_4$ with each $G_j$ the corresponding causal graph depicted in Figure 3. On the other hand, under our current definition, program $P_{15}$ has a unique causal stable model satisfying $I_{15}(d) = G_1' + G_2'$ with $G_1$ and $G_2$ being the result of removing every edge containing an atom label $A \in At$ in the causal graphs $G_1'$ and $G_2'$, respectively. Note that causal graphs $G_3$ and $G_4$ can be considered somehow redundant with respect to $G_1$ and $G_2$ and, in this sense, our current definition, not only extends the definition of causal stable models given in (Cabalar et al. 2014a) for disjunctive logic programs, but also better captures the notion of non-redundant justification for non-disjunctive programs (for more details we refer to Cabalar and Fandinno 2016b).

Papers on reasoning about actions and change (Lin 1995; McCain 1997; Thielscher 1997) have been traditionally focused on using causal inference to solve representational problems (mostly, the frame, ramification and qualification problems) without paying much attention to the derivation of cause-effect relations. Perhaps the most established AI approach for causality is relying on *causal networks* (Pearl 2000). In this approach, it is possible to conclude cause-effect relations like "$A$ has caused $B$" from the behaviour of structural equations by applying the counterfactual interpretation: "had $A$ not happened, $B$ would not have happened." As discussed by Hall (2004), this counterfactual-based definition corresponds to recognising some kind of *dependence* relation in the behaviour of a non-causal system description. As opposed to this, Hall considers a different (and incompatible) definition where causes must be connected to their effects via *sequences of causal intermediates*, something that is closer to our explanations in terms of causal graphs. A similar approach has also been studied by Vennekens (2011) in CP-logic.

Focusing on LP, our work obviously relates to approaches on justifications (Specht 1993; Pemmasani et al. 2004; Pontelli et al. 2009; Damásio et al. 2013; Denecker et al. 2015; Schulz and Toni 2016) which, as mentioned in the introduction, are limited to non-disjunctive programs. Among these, Why-not Provenance Justifications (WnP) (Damásio et al. 2013) share with our approach a semantic definition in terms of algebraic operations. A formal comparison was done in (Cabalar and Fandinno 2016a). With respect to (Pontelli et al. 2009), a formal relation has

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

not been established yet. We conjecture that our causal explanations can be seen as non-redundant off-line justifications with respect to a kind of "maximal set of assumptions." Still, this would apply to non-disjunctive programs only, since (Pontelli et al. 2009) do not cover disjunction.

A more far-fetched resemblance exists with respect to other meta-programming techniques for ASP debugging (Gebser et al. 2008; Oetsch et al. 2010) which have a different goal, tackling the question of why some interpretations are not stable models of a (possibly disjunctive) program without negative literals in the head. These approaches can be used to generate explanations for a set of literals $L$, with respect to an interpretation $I$, by recursively asking why $I \setminus L$ is not a stable model of the program. However, the given explanation cannot be used for our purpose of identifying causal relations, since it just provides a plain set of rules, without distinguishing which of them are a part of the same justification or which are their dependences.

## 6 Conclusions and open issues

We have provided an extension of a logic programming semantics based on causal justifications to cope with disjunctive programs. As in the previous, non-disjunctive approach, each true atom is assigned a set of non-redundant justifications, expressions built with rule labels and three algebraic operations: addition, product and application. We have shown that, for each standard stable model of a program, there is a class of causal stable models that capture the different ways in which (the same) true atoms can be justified. We have also shown that disjunctive rules are not enough to capture informal statements like "$A$ may cause $B$" and introduced causal-choice rules to represent them.

Regarding complexity, since the existence of causal stable models is completely determined by the existence of standard stable models, it intermediately follows that this problem is $\Sigma_2^{\mathrm{P}}$-complete (Eiter and Gottlob 1995). For query answering, it has been shown in (Cabalar et al. 2014b), that deciding whether a causal graph is a brave necessary cause is $\Sigma_2^{\mathrm{P}}$-complete, even in the non-disjunctive case. Whether disjunction makes the problem harder is still an open question.

Several other topics remain open for future study. Interesting topics include a complexity assessment or studying an extension to arbitrary theories as with Equilibrium Logic (Pearce 2006) for the non-causal case. Further ongoing work is focused on implementation, the introduction of strong negation, a formal comparative with (Pontelli et al. 2009) and (Schulz and Toni 2016).

## References

Baral, C. 2003. *Knowledge representation, reasoning and declarative problem solving*. Cambridge university press.

Cabalar, P. and Fandinno, J. 2016a. Enablers and inhibitors in causal justifications of logic programs. *Theory and Practice of Logic Programming (TPLP), (First View)*, 1–26.

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

Cabalar, P. and Fandinno, J. 2016b. Justifications for programs with disjunctive and causal-choice rules. *CoRR abs/1608.00870*.

Cabalar, P., Fandinno, J. and Fink, M. 2014a. Causal graph justifications of logic programs. *Theory and Practice of Logic Programming (TPLP) 14,* 4-5, 603–618.

Cabalar, P., Fandinno, J. and Fink, M. 2014b. A complexity assessment for queries involving sufficient and necessary causes. In *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proc.*, E. Fermé and J. Leite, Eds. Lecture Notes in Computer Science, vol. 8761. Springer, 297–310.

Damásio, C. V., Analyti, A. and Antoniou, G. 2013. Justifications for logic programming. In *Logic Programming and Nonmonotonic Reasoning, Twelfth Intl. Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proc.*, P. Cabalar and T. C. Son, Eds. Lecture Notes in Computer Science, vol. 8148. Springer, 530–542.

Denecker, M., Brewka, G. and Strass, H. 2015. A formal theory of justifications. In *Logic Programming and Nonmonotonic Reasoning - 13th Intl. Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proc.*, F. Calimeri, G. Ianni, and M. Truszczynski, Eds. Lecture Notes in Computer Science, vol. 9345. Springer, 250–264.

Eiter, T. and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence 15,* 3-4, 289–323.

Fandinno, J. 2015a. A causal semantics for logic programming. Ph.D. thesis, University of Corunna.

Fandinno, J. 2015b. Towards deriving conclusions from cause-effect relations. In *in Proc. of the 8th Intl. Workshop on Answer Set Programming and Other Computing Paradigms, ASPOCP 2015, Cork, Ireland, August 31, 2015*.

Fandinno, J. 2016. Deriving conclusions from non-monotonic cause-effect relations. *Theory and Practice of Logic Programming (TPLP)*. (to appear).

Gebser, M., Pührer, J., Schaub, T. and Tompits, H. 2008. A meta-programming technique for debugging answer-set programs. In *Proc. of the 23rd AAAI Conf. on Artificial Intelligence, Chicago, USA, July 13-17, 2008*, D. Fox and C. P. Gomes, Eds. AAAI Press, 448–453.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Logic Programming, Proc. of the Fifth Intl. Conference and Symposium, Seattle, Washington, August 15-19*, R. A. Kowalski and K. A. Bowen, Eds. MIT Press, 1070–1080.

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

Hall, N. 2004. Two concepts of causation. In *Causation and counterfactuals*, J. Collins, N. Hall, and L. A. Paul, Eds. Cambridge, MA: MIT Press, 225–276.

Hitchcock, C. and Knobe, J. 2009. Cause and norm. *Journal of Philosophy 11*, 587–612.

Lewis, D. K. 1973. Causation. *The Journal of Philosophy 70,* 17, 556–567.

Lin, F. 1995. Embracing causality in specifying the indirect effects of actions. In *Proc. of the Fourteenth Intl. Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*. Morgan Kaufmann, 1985–1993.

Marek, V. W. and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm*, K. R. Apt, V. W. Marek, M. Truszczyński, and D. Warren, Eds. Artificial Intelligence. Springer, 375–398.

McCain, N. C. 1997. Causality in commonsense reasoning about actions. Ph.D. thesis, University of Texas at Austin.

McCarthy, J. 1987. Epistemological problems of artificial intelligence. *Readings in artificial intelligence*, 459.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3-4, 241–273.

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press

Oetsch, J., Pührer, J. and Tompits, H. 2010. Catching the ouroboros: On debugging non-ground answer-set programs. *CoRR abs/1007.4986*.

Pearce, D. 2006. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence 47,* 1-2, 3–41.

Pearl, J. 2000. *Causality: models, reasoning, and inference*. Cambridge University Press, New York, NY, USA.

Pemmasani, G., Guo, H., Dong, Y., Ramakrishnan, C. R., and Ramakrishnan, I. V. 2004. Online justification for tabled logic programs. In *Functional and Logic Programming, 7th Intl. Symposium, FLOPS 2004, Nara, Japan, April 7-9, 2004, Proc*., Y. Kameyama and P. J. Stuckey, Eds. Lecture Notes in Computer Science, vol. 2998. Springer, 24–38.

Pontelli, E., Son, T. C. and El-Khatib, O. 2009. Justifications for logic programs under answer set semantics. *Theory and Practice of Logic Programming (TPLP) 9,* 1, 1–56.

Schulz, C. and Toni, F. 2016. Justifying answer sets using argumentation. *Theory and Practice of Logic Programming (TPLP) 16,* 1, 59–110.

Specht, G. 1993. Generating explanation trees even for negations in deductive database systems. In *Proc. of the 5th Workshop on Logic Programming Environments (LPE 1993), October 29-30, 1993, Vancouver, British Columbia, Canada*, M. Ducassé, B. L. Charlier, Y. Lin, and L. Ü. Yalçinalp, Eds. IRISA, Campus de Beaulieu, France, 8–13.

Thielscher, M. 1997. Ramification and causality. *Artificial Intelligence 89,* 1-2, 317–364.

Vennekens, J. 2011. Actual causation in CP-logic. *Theory and Practice of Logic Programming (TPLP) 11,* 4-5, 647–662.

https://doi.org/10.1017/S1471068416000454 Published online by Cambridge University Press