Deriving Conclusions From Non-Monotonic Cause-Effect Relations

Jorge Fandinno

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

Abstract

We present an extension of Logic Programming (under stable models semantics) that, not only allows concluding whether a true atom is a cause of another atom, but also *deriving new conclusions* from these causal-effect relations. This is expressive enough to capture informal rules like "if some agent's actions $\mathcal{A}$ have been *necessary* to cause an event E then conclude atom $caused(\mathcal{A}, E)$," something that, to the best of our knowledge, had not been formalised in the literature. To this aim, we start from a first attempt that proposed extending the syntax of logic programs with so-called *causal literals*. These causal literals are expressions that can be used in rule bodies and allow inspecting the derivation of some atom A in the program with respect to some query function ψ. Depending on how these query functions are defined, we can model different types of causal relations such as sufficient, necessary or contributory causes, for instance. The initial approach was specifically focused on monotonic query functions. This was enough to cover sufficient cause-effect relations but, unfortunately, necessary and contributory are essentially *non-monotonic*. In this work, we define a semantics for non-monotonic causal literals showing that, not only extends the stable model semantics for normal logic programs, but also preserves many of its usual desirable properties for the extended syntax. Using this new semantics, we provide precise definitions of *necessary* and *contributory* causal relations and briefly explain their behaviour on a pair of typical examples from the Knowledge Representation literature. (Under consideration for publication in Theory and Practice of Logic Programming)

1 Introduction

An important difference between classical models and most Logic Programming (LP) semantics is that, in the latter, true atoms must be founded or justified by a given derivation. Consequently, falsity is understood as absence of proof: for instance, a common informal way of reading for default literal *not A* is "there is no way to derive A." Although this idea seems quite intuitive and, in fact, several approaches have studied how to syntactically build these derivations or *justifications* (Specht 1993; Pemmasani et al. 2004; Pontelli et al. 2009; Denecker et al. 2015; Schulz and Toni 2016), it actually resorts to a concept, the *ways to derive* A, outside the scope of the standard LP semantics.

Such information on justifications for atoms can be of great interest for Knowledge Representation (KR), and especially, for dealing with problems related to causality. For instance, in the area of legal reasoning where determining a legal responsibility usually involves finding out which agent or agents have eventually caused a given result, regardless the chain of effects involved in the process. In this sense, an important challenge in causal reasoning is the capability of not only deriving facts of the form "A has caused B," but also being able to represent and

reason about them. As an example, take the assertion:

$$\text{"If somebody causes an accident, (s)he would receive a fine"} \tag{1}$$

This law does not specify the possible ways in which a person may cause an accident. Depending on a representation of the domain, the chain of events from the agent's action(s) to the final effect may be simple (a direct effect) or involve a complex set of indirect effects and defaults like inertia. Focussing on representing (1) in an elaboration tolerant manner (McCarthy 1998), we should be able to write a single rule whose body only refers to the $agent$ involved and the $accident$. For instance, consider the following program

$$accident \leftarrow oil \tag{2}$$
$$oil \qquad \leftarrow suzy \tag{3}$$
$$suzy \tag{4}$$

representing that $accident$ is an indirect effect of Suzy's actions. We may then represent (1) by the following rule

$$fine(suzy) \leftarrow suzy \text{ necessary for } accident \tag{5}$$

that states that Suzy would receive a $fine$ whenever the fact $suzy$ was necessary to cause the atom $accident$.

With this long term goal in mind, (Cabalar et al. 2014a) proposed a multi-valued semantics for LP that extends the stable model semantics (Gelfond and Lifschitz 1988) and where justifications are treated as *algebraic* constructions. In this semantics, *causal stable models* assign, to each atom, one of these algebraic expressions that captures the set of all non-redundant logical proofs for that atom. Recently, this semantics was used in (Fandinno 2015b) to extend the syntax of logic programs with a new kind of literal, called *causal literal*, that allow representing rules like

$$fine(suzy) \leftarrow suzy \text{ sufficient for } accident \tag{6}$$

and derive, from a program P_1 containing rules (2-4,6), that $fine(suzy)$ holds. However, the major limitation of this semantics is that causal literals must be monotonic and, therefore, rule (5) cannot be represented. It is easy to see that rule (5) is non-monotonic: in a program P_2 containing rules (2-5), the fact $suzy$ is necessary for $accident$ is satisfied and, thus, $fine(suzy)$ must hold, but in a program P_3 obtained by adding a fact oil to this last program, $suzy$ is not longer necessary and, thus, $fine(suzy)$ should not be a conclusion.

In this paper, we present a semantics for logic programs with causal literals defined in terms of *non-monotonic* query functions. More specifically, we summarise our contributions as follows. In Section 2, we define the syntax of causal literals and a multi-valued semantics for logic programs whose causal values rely on a completely distributive lattice based on causal graphs. Section 3 shows that positive monotonic program has a least model that can be computed by an extension of the direct consequences operator (van Emden and Kowalski 1976). In Section 4, we define semantics for programs with negation and non-monotonic causal literals and show that it is a conservative extension of the standard stable model semantics. Besides, with a running example, we show how causal literals can be used to derive new conclusion from necessary causal relations and, in Section 5, briefly relate this notion with the actual cause literature. In this section, we also formalise the weaker notion of *contributory cause*, also related to the actual cause literature, and show how causal literals may be used to derive new conclusion from them. In Section 6, we show that our semantics satisfy the usual properties of the stable modles semantics for the new

syntax. Finally, Section 7 concluded the paper. The online appendices include the definition of our semantics with nested expression in the body, the formal relation with (Fandinno 2015b), the proof of formal results from the paper and the formalisation of a Splitting Theorem for causal programs analgous to (Lifschitz and Turner 1994).

2 Causal Programs

We start by reviewing some definitions from (Cabalar et al. 2014a).

Definition 1 (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 possible infinite) set of terms. ☐

When $S = \{t_1, \ldots, t_n\}$ is a finite set, we will write $t_1 * \ldots * t_n$ and $t_1 + \ldots + t_n$ instead of $\prod S$ and $\sum S$, respectively. When $S = \emptyset$, we denote $\prod S$ and $\sum S$ by 1 and 0, respectively. We assume that application '$\cdot$' has higher priority than product '$*$' and, in its turn, product '$*$' has higher priority than addition '$+$'. *Application* '$\cdot$' represents application of a rule label to a previous justifications. For instance, the justification in program P_1 for atom $suzy$ is the fact $suzy$ itself. If rules (2-3) in program P_1 are labelled in the following way

$$r_1 : \quad accident \leftarrow oil \tag{7}$$
$$r_2 : \quad oil \quad\quad \leftarrow suzy \tag{8}$$

we may represent the justification of oil as $suzy{\cdot}r_2$, in other words, oil is true because of the the application of rule r_2 to the fact $suzy$. Similarly, we may represent the justification of $accident$ as $suzy{\cdot}r_2{\cdot}r_1$. Addition '$+$' is used to capture alternative independent causes: each addend is one of those independent causes. For instance, the justification of oil, in program P_3, may be represented as $suzy{\cdot}r_2 + oil$ and the justification of $accident$ as $(suzy{\cdot}r_2 + oil) \cdot r_1$. As we will see below application distributes over addition, so that, the justification of $accident$ can also be written as $suzy{\cdot}r_2{\cdot}r_1 + oil{\cdot}r_1$, which better illustrates the existence of two alternatives. Product '$*$' represents conjunction or joint causation. For instance, in a program P_4 obtained by adding the fact $billy$ to P_3 and replacing rule (8) by

$$r_2 : \quad oil \leftarrow suzy,\ billy \tag{9}$$

the justifications of oil will be $(suzy * billy){\cdot}r_2 + oil$. Similarly, the justification of $accident$ will be $(suzy * billy){\cdot}r_2{\cdot}r_1 + oil{\cdot}r_1$. Intuitively, terms without addition '$+$' represent individual causes while terms with addition '$+$' represent sets of causes. It is worth to mention that these algebraic expressions are in a one-to-one correspondence with non-redundant proofs of an atom (Cabalar et al. 2014a) and that they may also be understood as a formalisation of Lewis' concept of causal chain (Lewis 1973) (see Fandinno 2015b).

Definition 2 (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 sums '$+$'.* ☐

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

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

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

All three operations, '$*$', '$+$' and '$\cdot$' 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:

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

An immediately consequence of this definition is that product, addition, 1 and 0 respectively are the greatest lower bound, the least upper bound and the top and the bottom element of the $\leq$-relation. 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 false. Furthermore, applying distributivity (and absorption) of product and application over addition, every term can be represented in *(minimal) disjunctive normal form* in which addition is not in the scope of any other operation and every pair of addends are pairwise $\leq$-incomparable. In the following, we will assume that every term is in disjunctive normal form.

This semantics was used in (Fandinno 2015b), to define the concept of causal query, here *m-query*: a monotonic function $\phi : \mathbf{C}_{Lb} \longrightarrow \{0, 1\}$. Unfortunately, m-queries are not expressive enough to capture necessary causation for two reasons: (i) they are monotonic and (ii) they cannot capture relations between sets of causes. We introduced here the following definition which removes these two limitations.

Definition 3 (Causal query). *A causal query $\psi : \mathbf{C}_{Lb} \times \mathbf{V}_{Lb} \longrightarrow \{0, 1\}$ is a function mapping pairs cause-value into 1 (true) and 0 (false) which is anti-monotonic in the second argument, that is, $\psi(G, t) \leq \psi(G, u)$ for every $G \in \mathbf{C}_{Lb}$ and $\{t, u\} \subseteq \mathbf{V}_{Lb}$ such that $t \geq u$.* ◻

Syntax. We define the semantics of logic programs using its grounding. Therefore, for the remainder of this paper, we restrict our attention to ground logic programs. A *signature* is a triple $\langle At, Lb, \Psi \rangle$ where At, Lb and Ψ respectively represent sets of *atoms* (or *propositions*), *labels* and causal queries. We assume the signature of every program contains a causal query $\psi^1 \in \Psi$ s.t. $\psi^1(G, t) \overset{\text{def}}{=} 1$ for every $G \in \mathbf{C}_{Lb}$ and value $t \in \mathbf{V}_{Lb}$.

Definition 4 (Causal literal). *A (causal) literal is an expression $(\psi :: A)$ where $A \in At$ is an atom and $\psi \in \Psi$ is a causal query.* ◻

A causal atom $(\psi^1 :: A)$ is said to be *regular* and, by abuse of notation, we will use atom A as shorthand for regular causal literals of the form $(\psi^1 :: A)$. We will see below the justification for this notation. A *literal* is either a causal literal $(\psi :: A)$ *(positive literal)*, or a negated causal literal *not*$(\psi :: A)$ *(negative literal)* or a double negated causal literal *not not*$(\psi :: A)$ *(consistent literal)* with $A \in At$ an atom and $\psi \in \Psi$ a causal query.

Definition 5 (Causal program). *A* (causal) *program P is a set of rules of the form:*

$$r_i : \quad A \leftarrow B_1, \dots, B_m \tag{10}$$

where $0 \leq m$ is a non-negative integer, $r_i \in Lb$ is a label or $r_i = 1$, A (the head *of the rule) is an atom and each B_i with $1 \leq i \leq m$ (the* body *of the rule) is a literal or a term.* □

A rule r is said to be *positive* iff all literals in its body are positive and it is said to be *regular* if all causal literals in its body are regular. When $m = 0$, we say that the rule is a *fact* and omit the body and sometimes the symbol '$\leftarrow$.' Furthermore, for clarity sake, we also assume that, for every atom $A \in At$, there is an homonymous label $A \in Lb$ and that the label of an unlabelled rule is assumed to be its head. In this sense, a fact A in a program actually stands for the labelled rule $(A : A \leftarrow)$. A program P is *positive* or *regular* when all its rules are positive (i.e. it contains no default negation) or regular, respectively. A *standard program* is a regular program in which the label of every rule is '1 :'.

Semantics. A *(causal) interpretation* is a mapping $I : At \longrightarrow \mathbf{V}_{Lb}$ assigning a value to each atom. For interpretations I and J, we write $I \leq J$ when $I(A) \leq J(A)$ for every atom $A \in At$. Hence, there is a $\leq$-bottom interpretation $\mathbf{0}$ (resp. a $\leq$-top interpretation $\mathbf{1}$) that stands for the interpretation mapping every atom A to 0 (resp. 1). For an interpretation I and atom $A \in At$, by $\max I(A)$ we denote the set

$$\max I(A) \;\overset{\text{def}}{=}\; \big\{\, G \in \mathbf{C}_{Lb} \;\big|\; G \leq I(A) \text{ and there is no } G' \in \mathbf{C}_{Lb} \text{ s.t. } G < G' \leq I(A) \,\big\}$$

containing the maximal terms without addition (or individual causes) of A w.r.t. I.

Definition 6 (Causal literal valuation). *The* valuation of a causal literal *of the form $(\psi :: A)$ with respect to an interpretation I, in symbols $I(\psi :: A)$, is given by*

$$I(\psi :: A) \;\overset{\text{def}}{=}\; \sum \big\{\, G \in \max I(A) \;\big|\; \psi(G, I(A)) = 1 \,\big\}$$

We say that I satisfies a causal literal $(\psi :: A)$, in symbols $I \models (\psi :: A)$, iff $I(\psi :: A) \neq 0$. □

Notice now that $I(\psi^1 :: A) = I(A)$ for any atom A and, thus, writing a standard atom A as a shorthand for causal literal $(\psi^1 :: A)$ does not modify its intended meaning. Causal literals can be used to represent the body of rule (5). For instance, given a set of labels $\mathcal{A} \subseteq Lb$ representing the actions of some agent $\mathcal{A}$, we may define the query function

$$\psi_{\mathcal{A}}^{\text{nec}}(G, t) \;\overset{\text{def}}{=}\; \begin{cases} 1 & \text{if } t \leq \sum \mathcal{A} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

and represent the body of rule (5) by a causal literal of the form $(\psi_{Suzy}^{\text{nec}} :: accident)$ where $Suzy$ is the set of labels $\{suzy\}$. In the sake of clarity, we usually will write $(\mathcal{A} \; \texttt{necessary for} \; A)$ in rule bodies instead $(\psi_{\mathcal{A}}^{\text{nec}} :: A)$.

If we consider an interpretation I which assigns to the atom $accident$ its justification in program P_2, that is, $I(accident) = suzy{\cdot}r_2{\cdot}r_1$, then any term without addition $G \in \mathbf{C}_{Lb}$, satisfies

$$
\begin{aligned}
\psi_{Suzy}^{\text{nec}}(G, I)(accident) = 1 \quad &\text{iff} \quad suzy{\cdot}r_2{\cdot}r_1 \;\leq\; \sum \{suzy\} \\
&\text{iff} \quad suzy{\cdot}r_2{\cdot}r_1 \;\leq\; suzy \\
&\text{iff} \quad suzy{\cdot}r_2{\cdot}r_1 + suzy \;=\; suzy
\end{aligned}
$$

which holds applying application identity, associativity and absorption w.r.t. addition

$$suzy{\cdot}r_2{\cdot}r_1 + suzy \;=\; 1 \cdot suzy \cdot (r_2{\cdot}r_1) + suzy \;=\; suzy$$

Similarly, in program P_3, $\psi^{\text{nec}}_{Suzy}(G, I'(accident)) = 1$ iff $suzy{\cdot}r_2{\cdot}r_1 + oil \leq suzy$ which does not hold. In other words, Suzy's actions has been necessary in program P_2 but not in program P_3.

The valuation of a causal term t is the class of equivalence of t. The valuation of non-positive literals is defined as follows

$$I(not(\psi :: A)) \quad \overset{\text{def}}{=} \quad \begin{cases} 1 & \text{iff } I(\psi :: A) \;=\; 0 \\ 0 & \text{otherwise} \end{cases}$$

$$I(not\,not(\psi :: A)) \quad \overset{\text{def}}{=} \quad \begin{cases} 1 & \text{iff } I(\psi :: A) \;\neq\; 0 \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, for any literal or term L, we write $I \models L$ iff $I(L) \neq 0$.

Definition 7 (Causal model). *Given a rule r of the form* (10)*, we say that an interpretation I satisfies r, in symbols $I \models r$, if and only if the following condition holds:*

$$\big(I(B_1) * \ldots * I(B_m) \big) \cdot r_i \;\leq\; I(A) \tag{12}$$

An interpretation I is a causal model *of P, in symbols $I \models P$, iff I satisfies all rules in P.* □

Let P_5 be the program containing rules (7) and (8) plus the labelled fact $(suzy : suzy \leftarrow)$ and P_6 be the program containing rules (7) and (9) plus the labelled facts $(suzy : suzy \leftarrow)$ and $(billy : billy \leftarrow)$. Then, it can be checked that these programs respectively have unique $\leq$-minimal models I_5 and I_6 which satisfy

$$I_5(accident) \;=\; suzy{\cdot}r_2{\cdot}r_1 \qquad\qquad I_6(accident) \;=\; (suzy * billy){\cdot}r_2{\cdot}r_1 + oil$$

Let now P_7 and P_8 be the labelled programs respectively obtained by adding the following rule

$$r_3 : \quad fine(suzy) \leftarrow suzy \ \texttt{necessary for} \ accident \tag{13}$$

(resulting of labelling rule (5) with r_3) to programs P_5 and P_6. Then it can be checked that these programs also have unique $\leq$-minimal models I_7 and I_8 which respectively agree with I_5 and I_6 in all atoms but in $fine(suzy)$ and, as we have seen above,

$$I_7(\psi^{\text{nec}}_{Suzy} :: accident) = I_7(accident) = suzy{\cdot}r_2{\cdot}r_1 \qquad I_8(\psi^{\text{nec}}_{Suzy} :: accident) = 0$$

Furthermore, by definition, it holds that $I_j(fine(suzy)) \;=\; I_j(\psi^{\text{nec}}_{Suzy} :: accident){\cdot}r_3$ for $j \in \{7,8\}$ which implies that

$$I_7(fine(suzy))) \;=\; suzy{\cdot}r_2{\cdot}r_3$$
$$I_8(fine(suzy))) \;=\; 0{\cdot}r_3 = 0$$

That is, Suzy would receive a fine for causing the accident, $I_7 \models fine(suzy)$, w.r.t P_7, but not w.r.t. program P_8 because $I_8 \not\models fine(suzy)$.

It is worth to note that positive programs may contain non-monotonic causal literals that, somehow, play the role of negation and, hence, they may have several $\leq$-minimal causal models. Consider, for instance, the following positive program P_9

$$r_1 : \quad p \qquad\qquad r_2 : \quad q \leftarrow \mathcal{A}_1 \ \texttt{necessary for} \ p$$

where $\mathcal{A}_1 \stackrel{\text{def}}{=} \{r_1\}$. Program P_9 has two $\leq$-minimal causal models. The first one which satisfies $I_9(p) = r_1$ and $I_9(q) = r_1 \cdot r_2$; and a second unintended one which satisfies $I_9'(p) = r_1 + r_2$ and $I_9'(q) = 0$. In the following section, we introduce the notion of *monotonic programs* which have a least model and a well-behaved direct consequences operator (when they are positive). In Section 4, we will see that, in fact, only I_9 is a causal stable model of program P_9.

3 Positive monotonic Programs

A causal query ψ is said to be *monotonic* iff $\psi(G, u) \leq \psi(G', w)$ for any values $\{G, G'\} \subseteq \mathbf{C}_{Lb}$ and $\{u, w\} \subseteq \mathbf{V}_{Lb}$ such that $G \leq G'$. A causal literal $(\psi :: A)$ is *monotonic* if ψ is monotonic. A program P is *monotonic* iff P all causal literals occurring in P are monotonic. We show next that every monotonic program can be reduced to the syntax and semantics of (Fandinno 2015b). For space reasons, we omit here the details of (Fandinno 2015b), which can be found in Appendix C.

Definition 8. *Given a query ψ (resp. m-query ϕ), its* corresponding m-query (resp. query) *is given by $\phi_\psi(G) \stackrel{\text{def}}{=} \psi(G, 1)$ (resp. $\psi_\phi(G, t) \stackrel{\text{def}}{=} \phi(G)$). Similarly, for any program P (resp. m-program Q) its* corresponding m-program Q (resp. program P) *is obtained by replacing every query ψ in P (resp. m-query ϕ in Q) by its corresponding m-query ϕ_ψ (resp.query ψ_ϕ).* □

Theorem 1. *If P is the corresponding program of some positive m-program Q with the syntax of Definition 5 or Q is the corresponding m-program of some positive monotonic program P, then an interpretation I is a model of P iff I is a model of Q.* □

An immediate consequence of Theorem 1, plus Theorem 3.8 in (Fandinno 2015b), is that positive monotonic programs have a least model that can be computed by iteration of the following extension of the direct consequences operator of van Emden and Kowalski (1976).

Definition 9 (Direct consequences)**.** *Given a causal program P, the operator of* direct consequences *is a function T_P from interpretations to interpretations such that*

$$T_P(I)(A) \stackrel{\text{def}}{=} \sum \left\{ \left(I(B_1) * \ldots * I(B_m) \right) \cdot r_1 \mid (r_i : A \leftarrow B_1, \ldots, B_m) \in P \right\}$$

for any interpretation I and any atom $A \in At$. The iterative procedure is defined as usual

$$
\begin{aligned}
T_P^{\uparrow \alpha}(\mathbf{0}) &\stackrel{\text{def}}{=} T_P(T_P^{\uparrow \alpha - 1}(\mathbf{0})) && \text{if } \alpha \text{ is a successor ordinal} \\
T_P^{\uparrow \alpha}(\mathbf{0}) &\stackrel{\text{def}}{=} \sum_{\beta < \alpha} T_P^{\uparrow \beta}(\mathbf{0}) && \text{if } \alpha \text{ is a limit ordinal}
\end{aligned}
$$

As usual 0 and ω respectively denote the first limit ordinal and the first limit ordinal that is greater than all integers. Thus, $T_P^{\uparrow 0}(\mathbf{0}) = \mathbf{0}$. □

Corollary 1. *Any (possibly infinite) positive monotonic program P has a least causal model I which (i) coincides with the least fixpoint $\mathrm{lfp}(T_P)$ of the direct consequences operator T_P and (ii) can be iteratively computed from the bottom interpretation $I = \mathrm{lfp}(T_P) = T_P^{\uparrow \omega}(\mathbf{0})$.* □

Corollary 1 guarantees that the least fixpoint of T_P is well-behaved and corresponds to the least model of the program P. In fact, we can check now that the least model I_6 of program P_6 satisfies $I_6(accident) = (suzy * billy) \cdot r_2 \cdot r_1 + oil \cdot r_1$. First note, that program P_6 contains facts $suzy, billy$ and oil whose label is the same as the name atom and, thus, $T_{P_6}^{\uparrow 1}(\mathbf{0})(A) = A$ for each atom $A \in \{suzy, billy, oil\}$. Then, since $T_{P_6}^{\uparrow 1}(\mathbf{0})(suzy) = suzy$, $T_{P_6}^{\uparrow 1}(\mathbf{0})(billy) = billy$ and

rule (8) and fact oil belong to program P_6, it follows that $T_{P_6}^{\uparrow 2}(\mathbf{0})(oil) = (suzy * billy) \cdot r_2 + oil$. Similarly, we can check that

$$T_{P_6}^{\uparrow 3}(\mathbf{0})(accident) = ((suzy * billy) \cdot r_2 + oil) \cdot r_1 = (suzy * billy) \cdot r_2 \cdot r_1 + oil \cdot r_1$$

and, thus, $I_6 = T_{P_6}^{\uparrow 3}(\mathbf{0})$ is the least fixpoint of T_{P_6}. Checking that $T_{P_5}^{\uparrow 3}(\mathbf{0}) = I_5$, that $T_{P_7}^{\uparrow 4}(\mathbf{0}) = I_7$ and that $T_{P_8}^{\uparrow 4}(\mathbf{0}) = I_8$ are the least fixpoint and the least models respectively of programs P_5, P_7 and P_8 is analogous.

It is easy to see that every true atom, according to the standard least model semantics, has a non-zero causal value associated in the causal least model of the program, that is, some associated cause. An interpretation I is *two-valued* when it maps each atom into the set $\{0, 1\}$. By I^{cl}, we denote the two-valued (or "classic") interpretation corresponding to some interpretation I s.t.

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

Corollary 2. *Let P be a regular, positive monotonic program and Q its standard unlabelled version obtained by removing all labels from the rules in P. Let I and J be the least models of P and Q, respectively. Then, $I^{cl} = J$.* $\qquad\square$

4 Non-monotonic causal queries and negation

We introduce now the semantics for programs with non-monotonic causal queries and negation by extending the concept of reduct (Gelfond and Lifschitz 1988) to causal queries.

Definition 10 (Reduct). *For any term t, by ψ^t we denote a query such that*

$$\psi^t(G, u) \;\stackrel{\text{def}}{=}\; \begin{cases} 1 & \text{iff exists some } G' \leq G \text{ s.t. } G' \in \max t \text{ and } \psi(G', t) = 1 \\ 0 & \text{otherwise} \end{cases}$$

The reduct *of a causal literal $(\psi :: A)$ w.r.t some interpretation I is itself if ψ is monotonic and $(\psi^{I(A)} :: A)$ if ψ is non-monotonic. The reduct of a program P w.r.t. an interpretation I, in symbols P^I, is the result of (i) removing all rules whose body contains a non satisfied negative or consistent literal, (ii) removing all the negative and consistent literals for the remaining rules and (iii) replacing the remaining causal literals $(\psi :: A)$ by their reducts $(\psi :: A)^I$.* $\qquad\square$

It is easy to see that the reduct P^I of any program P is a positive monotonic program and, therefore, it has a least causal model.

Definition 11 (Causal stable model). *We say that an interpretation I is a* causal stable model *of a program P iff I is the least model of the positive program P^I.* $\qquad\square$

We can check now that interpretation I_9 is, in fact, the unique causal stable model of program P_9. Let $Q = P_9^{I_9}$ be the reduct of program P_9 w.r.t. I_9 consisting in the following rules

$$r_1: \quad p \qquad\qquad r_2: \quad q \leftarrow (\psi :: p)$$

where $\psi(G, t) = 1$ iff there exists some $G' \leq G$ s.t. $G' \in \max I_9(p) = r_1$ and $\psi_{\mathcal{A}_1}^{\text{nec}}(G', I_9(p))$ iff $r_1 \leq G$ and $r_1 \leq \sum \mathcal{A}_1 = r_1$ iff $r_1 \leq G$. First note that $T_Q^{\uparrow \alpha}(\mathbf{0})(p) = r_1 = I_9(p)$ for any ordinal $\alpha \geq 1$ because r_1 is the only rule with the atom p in the head. Then, note that

$T_Q^{\uparrow\alpha}(\mathbf{0})(\psi :: p) = T_Q^{\uparrow\alpha}(\mathbf{0})(p)$ because $r_1 \leq G$ for every $G \in \max T_Q^{\uparrow\alpha}(\mathbf{0})(p) = r_1$ (there is only one such $G = r_1$) and, thus,

$$T_Q^{\uparrow\beta}(\mathbf{0})(q) \;=\; T_Q^{\uparrow\alpha}(\mathbf{0})(\psi :: p){\cdot}r_2 \;=\; T_Q^{\uparrow\alpha}(\mathbf{0})(p){\cdot}r_2 \;=\; r_1{\cdot}r_2 \;=\; I_9(q)$$

for any ordinal $\beta \geq 2$. Hence, I_9 is a causal stable model of P_9. On the other hand, we can check that I_9' is not a causal stable model of P_9. Let $Q' = P_9^{I_9'}$ be the reduct of program P_9 w.r.t. I_9' consisting in the same rules than program Q, but replacing ψ by ψ' where $\psi'(G, t) = 1$ iff there exists some $G' \leq G$ s.t. $G' \in \max I_9'(p) = r_1 + r_2$ and $\psi_{\mathcal{A}_1}^{\mathrm{nec}}(G', I_9'(p))$. As above, $T_{Q'}^{\uparrow\alpha}(\mathbf{0})(p) = r_1 \neq I_9'(p) = r_1 + r_2$ for any ordinal $\alpha \geq 1$ and, therefore, I_9 is not a causal stable model of program P_9.

It is worth to mention that, as happened with positive programs, we can stablish a correspondence between the causal stable models of regular programs and the standard stable models of their standard version.

Definition 12 (Two-valued equivalence)**.** *Two programs P and Q are said to be* two-valued equivalent *iff for every causal stable model I of P there is an unique causal stable model J of Q such that $I^{cl} = J^{cl}$, and vice-versa.* $\qquad\square$

Theorem 2. *Let P be a regular program and Q be its corresponding standard program obtained by removing all labels in P. Then P and Q are two-valued equivalent.* $\qquad\square$

Theorem 2 asserts that, labelling a standard program does not change which atoms are true or false in its stable models, in other words, the causal stable semantics presented here is a conservative extension of the standard stable model semantic.

5 Contributory cause and its relation with actual causation

Until now we have considered that an agent is a cause of an event when its actions have been necessary to cause that event. This understanding is similar to the definition of the *modified Halpern-Pearl definition of causality* given by Halpern (2015). However, in some scenarios it makes sense to consider a weaker definition in which those agents whose actions have *contributed* to that event are also considered causes, even if their actions have not been necessary (Pearl 2000). Consider, for instance, the following example from (Hopkins and Pearl 2003).

Example 1. *For a firing squad consisting of shooters Billy and Suzy, it is John's job to load Suzy's gun. Billy loads and fires his own gun. On a given day, John loads Suzy's gun. When the time comes, Suzy and Billy shoot the prisoner. The agents who caused the prisoner death would be punished with imprisonment.* $\qquad\square$

In this example, although the actions of any of the agents are not necessary for the prisoner's death, commonsense tells that all three should be considered responsible of it. If we represent Example 1 by the following program P_{10}

$$
\begin{array}{llll}
r_1 : & dead & \leftarrow shoot(suzy), loaded & shoot(suzy) \\
r_2 : & dead & \leftarrow shoot(billy) & shoot(billy) \\
r_3 : & loaded & \leftarrow load(john) & load(john) \\
r_{\mathcal{A}} : & long\_prison(\mathcal{A}) \leftarrow & \mathcal{A} \;\texttt{necessary for}\; dead
\end{array}
$$

for $\mathcal{A} \in \{suzy,\ billy,\ john\}$, it can be shown that its unique causal stable model I_{10} satisfies

$$I_{10}(dead) \;=\; \big(load(john){\cdot}r_3 * shoot(suzy)\big) \cdot r_1 \;+\; shoot(billy){\cdot}r_2$$

Recall that, we assume that every fact has a label with the same name. According to I_{10}, the actions of the three agents appear in the causes of the atom $dead$, but there is no agent whose actions occur in all causes. Then, the causal literal ($\mathcal{A}$ `necessary for` $dead$) is not satisfied for any agent $\mathcal{A}$ and, therefore, it holds that $I_{10}(long\_prison(\mathcal{A})) = 0$ for every agent $\mathcal{A} \in \{suzy,\ billy,\ john\}$. That is, no agent is punished with imprisonment for the prisoner's death. On the other hand, if P_{11} is a program obtained by replacing rules $r_{\mathcal{A}}$ by rules

$$c_{\mathcal{A}} : \; short\_prison(\mathcal{A}, dead) \leftarrow \mathcal{A} \text{ contributed to } dead$$

in program P_{10}, we may expect that $short\_prison(\mathcal{A})$ holds, in its unique causal stable model I_{11}, for any $\mathcal{A} \in \{suzy,\ billy,\ john\}$. We formalise this by defining the following query

$$\psi_{\mathcal{A}}^{\text{cont}}(G, t) \;\stackrel{\text{def}}{=}\; \begin{cases} 1 & \text{if } G \leq \sum \mathcal{A} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

In the sake of clarity, we will write ($\mathcal{A}$ `contributed to` $dead$) instead of ($\psi_{\mathcal{A}}^{\text{cont}} :: dead$). It can be checked that $\big(load(john){\cdot}r_3 * shoot(suzy)\big) \cdot r_1 \leq load(john)$ and, therefore,

$$I_{11}(john \text{ contributed to } dead) \;=\; \big(load(john){\cdot}r_3 * shoot(suzy)\big) \cdot r_1$$

Consequently, $I_{11}(short\_prison(john)) = \big(load(john){\cdot}r_3 * shoot(suzy)\big) \cdot r_1 \cdot c_{john}$. Similarly, it can be shown that

$$I_{11}(short\_prison(suzy)) \;=\; \big(load(john){\cdot}r_3 * shoot(suzy)\big) \cdot r_1{\cdot}c_{suzy}$$
$$I_{11}(short\_prison(billy)) \;=\; shoot(billy){\cdot}r_2 \cdot c_{billy}$$

It is worth to note that contributory causes are non-monotonic when defaults are taken into account. Consider now the following variation of Example 1.

Example 2. *Now Suzy also loads her gun as Billy does. However, Suzy's gun was broken and John repaired it.* □

As in Example 1, John's repairing action is necessary in order for Suzy to be able to fire her gun. However, in this case, it seems too severe to consider that John has contributed to the prisoner's death. This consideration has been widely attributed to the fact that we consider that, by default, things are not broken and that causes must be events that deviate from the norm (Maudlin 2004; Hall 2007; Halpern 2008; Hitchcock and Knobe 2009). If we represent this variation by a program P_{12} containing the following rules[1]

r_1 :	$dead$	$\leftarrow shoot(suzy), un\_broken$	$shoot(suzy)$
r_2 :	$dead$	$\leftarrow shoot(billy)$	$shoot(billy)$
r_3 :	un_broken	$\leftarrow repair(john)$	$repair(john)$
$c_{\mathcal{A}}$:	$short\_prison(\mathcal{A}) \leftarrow \mathcal{A}$ `contributed to` $dead$		

---

[1] We have chosen this representation in order to illustrate the non-monotonicity of contributory cause. However, solving the Frame and Qualification Problems (McCarthy and Hayes 1969; McCarthy 1987) would require the introduction of time and the inertia laws, plus the replacement of rule r_1 by the pair of rules ($r_1 : dead \leftarrow shoot(suzy),\ not\ ab$) and ($ab \leftarrow broken$). For a detailed discussion of how causality and the inertia laws can combined we refer to (Fandinno 2015a).

for $\mathcal{A} \in \{suzy,\ billy,\ john\}$, then it is easy to see that

$$I_{12}(dead) \;=\; \big(repair(john){\cdot}r_3 * shoot(suzy)\big) \cdot r_1 \;+\; shoot(billy){\cdot}r_2$$

where I_{12} is the least model of program P_{12} and, thus, $responsible(john, dead)$ will be a conclusion of it. Just note that program P_{12} is the result of replacing atoms $loaded$ and $load(john)$ in program P_{11} by un_broken and $repair(john)$, respectively. Note also that nothing in program P_{12} reflects the fact that by default guns are un_broken. We state that guns are un_broken by default adding the following rule

$$1: \quad un\_broken \leftarrow not\, broken \tag{15}$$

If P_{13} is the result of adding rule (15) to program P_{12} and I_{13} is the least model of P_{13}, then

$$I_{13}(un\_broken) \;=\; I_{12}(un\_broken) \;+\; 1 \;=\; 1$$

and, consequently,

$$
\begin{aligned}
I_{12}(dead) \;&=\; \big(1{\cdot}r_3 * shoot(suzy)\big) \cdot r_1 \;+\; shoot(billy){\cdot}r_2 \\
&=\; \big(r_3 * shoot(suzy)\big) \cdot r_1 \;+\; shoot(billy){\cdot}r_2
\end{aligned}
$$

which shows that John is not considered to have contributed to the prisoner's death. Hence, $short\_prison(john)$ is not a conclusion of program P_{13}. It is worth to mention that besides the two syntactic differences between causal queries and m-queries already mentioned, there is a, perhaps, less noticeable difference in the evaluation of causal literals. Note that,

$$\big(repair(john){\cdot}r_3 * shoot(suzy)\big) \cdot r_1 \;\leq\; \big(r_3 * shoot(suzy)\big) \cdot r_1$$

and, thus, if we replaced $G \in \max I(A)$ by $G \leq I(A)$ in Definition 6 (as done in Fandinno 2015b), it would follows that atom $short\_prison(john)$ would be an unintended conclusion of program P_{13}. It is also worth to mention that, besides (Pearl 2000) approach, the notion of contributory cause is also behind the definitions of actual cause given in (Halpern and Pearl 2005; Hall 2007).

6 Properties of causal logic programs

Theorem 2 established a correspondence for regular programs, but they say nothing about programs with causal queries. For instance, positive program with non-monotonic causal literals may have more than one causal stable model. Consider the following positive program P_{14}

$$
\begin{array}{llll}
r_1: & p & \qquad r_2: & q \leftarrow \mathcal{A}_1 \ \texttt{necessary for } p \\
r_3: & q & \qquad r_4: & p \leftarrow \mathcal{A}_2 \ \texttt{necessary for } q
\end{array}
$$

obtained by adding rules r_3 and r_4 to program P_9 and where $\mathcal{A}_2 \stackrel{\text{def}}{=} \{r_3\}$. Program P_{14} has two causal stable causal models. The first that satisfies $I_{14}(p) = r_1 + r_3{\cdot}r_4$ and $I_{14}(q) = r_3$. The second $I'_{14}(p) = r_1$ and $I'_{14}(q) = r_3 + r_1{\cdot}r_2$. Let now $Q = P_{14}^{I_{14}}$ be the reduct of program P_{14} w.r.t. I_{14}, which consists in the following rules

$$
\begin{array}{llll}
r_1: & p & \qquad r_2: & q \leftarrow (\psi_1 :: p) \\
r_3: & q & \qquad r_4: & p \leftarrow (\psi_2 :: q)
\end{array}
$$

where $\psi_1(G,t) = 1$ iff there exists some $G' \leq G$ such that $G' \in \max I_{14}(p) = r_1 + r_3{\cdot}r_4$ and $\psi_{\mathcal{A}_1}^{\text{nec}}(G', I_{14}(p))$ and $\psi_2(G,t) = 1$ iff there exists $G' \leq G$ such that $G' \in \max I_{14}(q) = r_3$

and $\psi_{\mathcal{A}_2}^{\text{nec}}(G', I_{14}(p))$. First, note that $\psi_{\mathcal{A}_1}^{\text{nec}}(G', I_{14}(p))$ iff $I_{14}(p) = r_1 + r_3 \cdot r_4 \leq \sum \mathcal{A}_1 = r_1$ which does not hold. Thus, $\psi_1(G, t) = 0$ for every $G \in \mathbf{C}_{Lb}$ and $t \in \mathbf{V}_{Lb}$. Then, it is clear that the body of rule r_2 is never satisfied and, therefore, $T_Q^{\uparrow\alpha}(\mathbf{0})(q) = r_3$ for any ordinal $\alpha \geq 1$. It can also be checked that $\psi_2(r_3, T_Q^{\uparrow\alpha}(\mathbf{0})(q)) = 1$ because there exists $G' = r_3$ such that $G' \in \max I'_{14}(q) = r_3$ and $\psi_{\mathcal{A}_2}^{\text{nec}}(G', I_{14}(q)) = \psi_{\mathcal{A}_2}^{\text{nec}}(r_3, r_3) = 1$ since $r_3 \leq \sum \mathcal{A}_2 = r_3$. Hence, since $r_3 \in \max T_Q^{\uparrow\alpha}(\mathbf{0})(q)$ and $\psi_2(r_3, T_Q^{\uparrow\alpha}(\mathbf{0})(q)) = 1$, it follows that $T_Q^{\uparrow\alpha}(\mathbf{0})(\psi_2 :: q) = r_3$ and $T_Q^{\uparrow\beta}(\mathbf{0})(p) = r_1 + T_Q^{\uparrow\alpha}(\mathbf{0})(q) \cdot r_4 = r_1 + r_3 \cdot r_4 = I_9(p)$ for any ordinal $\beta \geq 2$. Hence, I_{14} is the least model of $P_{14}^{I_{14}}$ and a causal stable model of program P_{14}. Showing that I'_{14} is also a causal stable model of P_{14} is symmetric.

In the following we revise some desired general properties for a LP semantics. First, causal stable models should also be supported models. Note that the concept of supported model bellow is analogous to the usual concept used in standard LP, but it is stronger in the sense that, not only requires that true atoms are supported, but also all their causes must be supported by a rule and a cause of its body.

Definition 13. *A interpretation I is a* (causally) supported model *of a program P iff I is a model of P and for every true atom A and cause $G \in \mathbf{C}_{Lb}$ such that $G \leq I(A)$ there is a rule r in P of the form of* (10) *such that $G \leq (I(B_1) * \ldots * I(B_m)) \cdot r_i$.* $\square$

Proposition 1. *Any causal stable model I of a program P is a also supported model of P.* $\square$

Furthermore, as happen with programs with nested negation under the standard stable models semantics (where stable models may not be minimal models of the program), causal stable models may not be minimal models either. In fact, this may happen even when the nested negation is replaced by a non-monotonic causal literal. Consider, for instance, the following program P_{15}

$$r_1: \quad p \qquad\qquad r_2: \quad p \leftarrow \mathit{not}\,(\mathcal{A}_1 \text{ necessary for } p)$$

where $\mathcal{A}_1 \stackrel{\text{def}}{=} \{r_1\}$. Program P_{15} has two causal models. One which satisfies $I_{15}(p) = r_1$. The other which satisfies $I'_{15}(p) = r_1 + r_2$. We define now the notion of *normal program* whose causal stable models are also $\leq$-minimal models. A program P is *normal* iff no body rule in P contains a consistent literal (double negated literal) nor a negated non-monotonic causal literal. In other words, a program is normal iff it does not contains nested negation nor non-monotonic causal literals in the scope of negation.

Proposition 2. *Any causal stable model I of normal program P is also a $\leq$-minimal model.* $\square$

Splitting programs. The intuitive meaning of the causal rule (13) in programs P_7 and P_8 is to cause the atom $fine(suzy)$ whenever the causal query expressed by its body is true with respect to a programs P_5 and P_6, respectively. This intuitive understanding can be formalised as a splitting theorem in (Lifschitz and Turner 1994).

Theorem 3 (Splitting). *Let $\langle P_b, P_t \rangle$ a partition of a program P such that no atom occurring in the head of a rule in P_t occurs in P_b. An interpretation I is a causal stable model of P iff there is some causal stable model J of P_b such that I is a causal stable model of $(J \cup P_t)$.* $\square$

In our running example, the bottom part are $P_{7,b} = P_5$ and $P_{8,b} = P_6$ while their top part $P_{7,t} = P_{8,t}$ is the program containing the rule (13). This result can be generalised to infinite splitting sequences as follows.

Definition 14. *A* splitting sequence *of a program P is a family $(P_\alpha)_{\alpha<\mu}$ of pairwise disjoint sets such that $P = \bigcup_{\alpha<\mu} P_\alpha$ and no atom occurring in the head of a rule in some P_α occurs in the body of a rule in $\bigcup_{\beta<\alpha} P_\beta$. A* solution *of a splitting $(P_\alpha)_{\alpha<\mu}$ is a family $(I_\alpha)_{\alpha<\mu}$ such that align=Center, leftmargin=10pt, itemindent=0.5pt*

1. *I_0 is a stable model of P_0,*

2. *I_α is a stable model of $(J_\alpha \cup P_\alpha)$ for any ordinal $0 < \alpha < \mu$ where $J_\alpha = \sum_{\beta<\alpha} I_\beta$.*

A splitting sequence is said to be strict in α *if, in addition, no atom occurring in the head of a rule in P_α occurs (in the head of a rule) in $\bigcup_{\beta<\alpha} P_\beta$ and it is said to be* strict *if it is strict in α for every $\alpha < \mu$.* □

Theorem 4 (Splitting sequences)**.** *Let $(P_\alpha)_{\alpha<\mu}$ a splitting sequence of some program P. An interpretation I is a causal stable model of P iff there is some solution $(I_\alpha)_{\alpha<\mu}$ of $(P_\alpha)_{\alpha<\mu}$ such that $I = \sum_{\alpha<\mu} I_\alpha$. Furthermore, if such solution is strict in α, then $I_\alpha = I_{|S_\alpha}$ where S_α is the set of all atoms not occurring in the head of any rule in $\bigcup_{\alpha<\beta<\mu} P_\beta$ and $I_{|S_\alpha}$ denotes the restriction if I to S_α.* □

A program P is said to be *stratified* if there is a some ordinal μ and mapping λ from the set of atoms At into the set of ordinals $\{\alpha < \mu\}$ such that, for every rule of the form (10) and atom B occurring in its body, it satisfies $\lambda(A) \geq \lambda(B)$ if B does not occur in the scope of negation nor in a non-monotonic causal literal, and $\lambda(A) > \lambda(B)$ if B does occur under the scope of negation or in a non-monotonic causal literal.

Proposition 3. *Every stratified causal program P has a unique causal stable model.* □

7 Conclusions, related work and open issues

The main contribution of this work is the introduction of a semantics for non-monotonic *causal literals* that allow deriving new conclusions by inspecting the causal justifications of atoms in an *elaboration tolerant* manner. In particular, we have used causal literals to define *necessary* and *contributory causal relations* which are intuitively related to some of the most established definitions of actual causation in the literature (Pearl 2000; Halpern and Pearl 2005; Hall 2007; Halpern 2015). Besides, by some running examples we have shown that causal literals allow, not only to derive whether some event is the cause or not of another event, but also to derive new conclusions from this fact. From a technical point of view, we have shown that our semantics is a conservative extension of the stable model semantics and that satisfy the usual desired properties for an LP semantics (casual stable models are supported models, minimal models in case of normal programs and can be iteratively computed by split table programs). It worth to mention that, besides the syntactic approaches to justifications in LP, the more related approach to our semantics is (Damásio et al. 2013), for which a formal comparative can be found in (Cabalar and Fandinno 2016a) and that (Pontelli et al. 2009) allows a Prolog system to reason about justifications of an ASP program, but justifications cannot be inspected inside the ASP program.

Regarding complexity, it has been shown in (Cabalar et al. 2014b) that there may be an exponential number of causes for a given atom w.r.t. each causal stable model. Despite that, the existence of stable model for programs containing only monotonic queries evaluable in polynomial time is NP-complete (Fandinno 2015b). For programs containing only necessary causal literals we can prove NP-complete (NP-hard holds even for programs containing a single negated

regular literal or positive programs containing a single constraint, see Proposition 35 in the Appendix). The complexity for programs including other non-monotonic causal literals (like contributory) is still an open question. A preliminary prototype extending the syntax of logic programs with causal literals capturing sufficient, necessary and contributory causal relation can be tested on-line at `http://kr.irlab.org/cgraphs-solver/nmsolver`.

In a companion paper (Cabalar and Fandinno 2016b), the causal semantics used here has been extended to disjunctive logic programs, which will be useful for representing non-deterministic causal laws. Interesting topics include a complexity assessment or studying an extension to arbitrary theories as with Equilibrium Logic (Pearce 2006) for the non-causal case; and formalise the relation between our notions of necessary and contributory cause with the above definitions of the actual causation and, in particular, with (Vennekens 2011) who has studied it in the context of CP-logic. A promising approach seems to translate structural equations into logic programs in a similar way as it has been done to translate them into the causal theories (Giunchiglia et al. 2004; Bochman and Lifschitz 2015).

References

BOCHMAN, A. AND LIFSCHITZ, V. 2015. Pearl's causality in a logical setting. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, B. Bonet and S. Koenig, Eds. AAAI Press, 1446–1452.

CABALAR, P. AND FANDINNO, J. 2016a. Enablers and inhibitors in causal justifications of logic programs. *Theory and Practice of Logic Programming, TPLP, (First View)*.

CABALAR, P. AND FANDINNO, J. 2016b. Justifications for programs with disjunctive and causal-choice rules. *Theory and Practice of Logic Programming TPLP*. (to appear).

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. Proceedings*, 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 International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, 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 International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings*, F. Calimeri, G. Ianni, and M. Truszczynski, Eds. Lecture Notes in Computer Science, vol. 9345. Springer, 250–264.

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 International Workshop on Answer Set Programming and Other Computing Paradigms, ASPOCP 2015, Cork, Ireland, August 31, 2015. Extended version under revison for publication in Fundamenta Informaticae*.

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, August 15-19*, R. A. Kowalski and K. A. Bowen, Eds. MIT Press, 1070–1080.

GIUNCHIGLIA, E., LEE, J., LIFSCHITZ, V., MCCAIN, N., AND TURNER, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence 153*, 1-2, 49–104.

HALL, N. 2007. Structural equations and causation. *Philosophical Studies 132*, 1, 109–136.

HALPERN, J. Y. 2008. Defaults and normality in causal structures. In *Principles of Knowledge Representa-*

tion and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008, G. Brewka and J. Lang, Eds. AAAI Press, 198–208.

HALPERN, J. Y. 2015. A modification of the halpern-pearl definition of causality. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Q. Yang and M. Wooldridge, Eds. AAAI Press, 3022–3033.

HALPERN, J. Y. AND PEARL, J. 2005. Causes and explanations: A structural-model approach. part I: Causes. *British Journal for Philosophy of Science 56,* 4, 843–887.

HITCHCOCK, C. AND KNOBE, J. 2009. Cause and norm. *Journal of Philosophy 11*, 587–612.

HOPKINS, M. AND PEARL, J. 2003. Clarifying the usage of structural models for commonsense causal reasoning. In *Proceedings of the AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*. 83–89.

LEWIS, D. K. 1973. Causation. *The journal of philosophy 70,* 17, 556–567.

LIFSCHITZ, V., TANG, L. R., AND TURNER, H. 1999. Nested expressions in logic programs. *Ann. Math. Artif. Intell. 25,* 3-4, 369–389.

LIFSCHITZ, V. AND TURNER, H. 1994. Splitting a logic program. In *Logic Programming, Proceedings of the Eleventh International Conference on Logic Programming, Santa Marherita Ligure, Italy, June 13-18, 1994*, P. V. Hentenryck, Ed. MIT Press, 23–37.

MAUDLIN, T. 2004. Causation, counterfactuals, and the third factor. In *Causation and Counterfactuals*, J. Collins, E. J. Hall, and L. A. Paul, Eds. MIT Press.

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

MCCARTHY, J. 1998. Elaboration tolerance. In *Proceedings of the Fourth Symposium on Logical Formalizations of Commonsense Reasoning, Common Sense*. London, UK, 198–217.

MCCARTHY, J. AND HAYES, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence Journal 4*, 463–512.

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, Seventh International Symposium, FLOPS 2004, Nara, Japan, April 7-9, 2004, Proceedings*, 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 *Proceedings of the Fifth Workshop on Logic Programming Environments (LPE 1993), October 29-30, 1993, In conjunction with ILPS 1993, Vancouver, British Columbia, Canada*, M. Ducassé, B. L. Charlier, Y. Lin, and L. Ü. Yalçinalp, Eds. IRISA, Campus de Beaulieu, France, 8–13.

VAN EMDEN, M. H. AND KOWALSKI, R. A. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM (JACM) 23,* 4, 733–742.

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

Appendix A. Nested expressions in rule bodies

In this section we extend the syntax presented in Section 2 in order to allow nested expressions in rule bodies (Lifschitz et al. 1999).

Definition 15. *A formula F is recursively defined as one of the following expressions*

$$F \quad ::= \quad t \ \mid \ C \ \mid \ E, H \ \mid \ E; H \ \mid \ \text{not } E$$

where t is a term, C is a causal literal (Definition 4) and both, E and H are formulas in their turn. □

A formula F is said to be *elementary* iff it is a term t or a causal literal C. It is said to be *regular* iff every causal literal occurring in it is regular and is said to be *positive* iff the operator *not* does not occur in it. F is said to *monotonic* iff every causal literal occurring in F is monotonic. In formulas, we will write $\top$ and $\bot$ instead of 1 and 0, respectively.

Definition 16 (Causal logic program)**.** *Given a signature $\langle At, Lb, \Psi \rangle$, a* (causal logic) program *P is a set of rules of the form:*

$$r_i : \quad A \ \leftarrow \ F \tag{A1}$$

where $r_i \in Lb$ is a label or $r_i = 1$, $A \in At$ (the head *of the rule) is an atom or $A = \bot$ and F (the* body *of the rule) is a formula.* □

A rule r is said to be *regular* iff its body is regular and its said to be *positive* iff its body is positive and $A \neq \bot$. It is said to be *monotonic* iff F is monotonic. If $F = \top$, we say the rule is a *fact* and omit the body and sometimes the symbol '$\leftarrow$.' A program P is *regular*, *positive* or *monotonic* when all its rules are regular, positive or monotonic, respectively. A *standard program* is a regular in which the label of every rule is '1 :'. Definition 16 extends Definition 5 by allowing nested expressions in the rule bodies. A causal program in the sense of Definition 5 is a program in which the body F of all rules are conjunctions of regular causal literals or their negation. Note that every rule of the form of (10) with $m = 0$ corresponds to a rule of the form of $(r_i : \ A \leftarrow \top)$.

Semantics. The semantics of causal logic programs with nested expressions is given as follows.

Definition 17 (Valuation)**.** *The valuation of causal literals and causal terms is as given by Definition 6. Otherwise, the valuation of a formula F is recursively defined as follows*

$$I(E, H) \ = \ I(E) * I(H) \qquad\qquad I(\text{not } E) \ = \ \begin{cases} 1 & \text{iff } I(E) = 0 \\ 0 & \text{otherwise} \end{cases}$$
$$I(E; H) \ = \ I(E) + I(H)$$

We say that I satisfies a formula F, in symbols $I \models F$, iff $I(F) \neq 0$. □

Definition 18 (Causal model)**.** *Given a rule r of the form* (A1)*, we say that an interpretation I satisfies r, in symbols $I \models r$, if and only if the following condition holds:*

$$I(F) \cdot r_i \ \leq \ I(A) \tag{A2}$$

An interpretation I is a causal model *of P, in symbols $I \models P$ iff I satisfies all rules in P.* □

The following result shows that Definition 18 agrees with Definition 7 for programs within the syntax of Definition 5 and, thus, the former is a conservative extension of the last to programs with nested expressions in the body.

Proposition 4. *For any program P with the syntax of Definition 5, an interpretation I is a model of P w.r.t. Definition 7 iff I is a model of P w.r.t. Definition 18.* $\square$

We also can extend the definition of the direct consequences operator to programs with nested expressions as follows.

Definition 19 (Direct consequences)**.** *Given a causal program with nested expressions P, the operator of* direct consequences *is a function T_P from interpretations to interpretations such that*

$$T_P(I)(A) \overset{\text{def}}{=} \sum \big\{ \ I(F) \cdot r_1 \ \big| \ (r_i : \ A \leftarrow F) \in P \ \big\}$$

for any interpretation I and any atom $A \in At$. The iterative procedure is defined as usual

$$
\begin{aligned}
T_P^{\uparrow\alpha}(\mathbf{0}) &\overset{\text{def}}{=} T_P(T_P^{\uparrow\alpha-1}(\mathbf{0})) && \text{if } \alpha \text{ is a successor ordinal} \\
T_P^{\uparrow\alpha}(\mathbf{0}) &\overset{\text{def}}{=} \sum_{\beta<\alpha} T_P^{\uparrow\beta}(\mathbf{0}) && \text{if } \alpha \text{ is a limit ordinal}
\end{aligned}
$$

As usual 0 and ω respectively denote the first limit ordinal and the first limit ordinal that is greater than all integers. Thus, $T_P^{\uparrow 0}(\mathbf{0}) = \mathbf{0}$. $\square$

We will show in the Appendix C that, if P is monotonic and positive, then the T_P operator has a least fixpoint that can be computed by iteration from the bottom interpretation $\mathbf{0}$.

Causal stable models of programs with nested expressions.

Definition 20 (Reduct)**.** *The reduct of a causal literal and terms is as in Definition 10. The reduct of formulas is inductively defined as follows*

$$
\begin{aligned}
(E, H)^I &= (E^I, H^I) \\
(E; H)^I &= (E^I; H^I)
\end{aligned}
\qquad
(\text{not } E)^I = \begin{cases} \bot & \text{if } I \models E^I \\ \top & \text{otherwise} \end{cases}
$$

The reduct of program P is the program

$$P^I \overset{\text{def}}{=} \{ \ r^I \ \big| \ r \in P \ \}$$

where the reduct r^I of a rule r like (10) is given by $(r_i : \ H \leftarrow F^I)$. $\square$

Definition 21 (Formula equivalence)**.** *A formula F is said to be* equivalent *to a formula E, in symbols $F \Leftrightarrow E$, iff any pair of causal interpretations I and J satisfy that $I(F^J) = I(E^J)$.*

Proposition 5. *For any formula F, the following simplifications are valid align=Center, left-margin=10pt, itemindent=0.5pt*

1. $(F, \top) \Leftrightarrow F$ and $(\top, F) \Leftrightarrow F$.
2. $(F; \top) \Leftrightarrow \top$ and $(\top; F) \Leftrightarrow \top$.
3. $(F, \bot) \Leftrightarrow \bot$ and $(\bot, F) \Leftrightarrow F$
4. $(F; \bot) \Leftrightarrow F$ and $(\bot; F) \Leftrightarrow F$. $\square$

Definition 22 (Causal stable model)**.** *We say that an interpretation I is a* causal stable model *of a program with nested expressions P iff I is the least model of the positive monotonic program P^I (Definition 20).* $\square$

Proposition 6. *For any program P with the syntax of Definition 5, the reduct of P w.r.t. to an interpretation I and Definition 10 is the same as the reduct of P w.r.t. I and Definition 20 after applying the simplifications from Proposition 5 and removing all rules whose body is $\bot$. Consequently, the causal stable models of P w.r.t. Definitions 11 and 22 are the same.* □

Proposition 7. *Let P be a causal program with nested expressions. Any causal stable model I of P is a model of P.* □

Proposition 8. *Let P be a causal program with nested expressions. Any causal stable model I of a P is a also supported model of P.* □

Proposition 9. *Let P be a causal program with nested expressions. Then, any causal stable model I of P is also a $\leq$-minimal model of P.* □

Note that Propositions 1 and 2 in the main part of the paper, are direct consequences of Proposition 6 together with Propositions 8 and 9, respectively.

Splitting programs. The intuitive meaning of the causal rule (13) in programs P_7 and P_8 is to cause the atom $responsible(suzy, accident)$ whenever the causal query expressed by its body is true with respect to a programs P_5 and P_6, respectively. This intuitive understanding can be formalised as a splitting theorem in (Lifschitz and Turner 1994).

Theorem 5 (Splitting)**.** *Let $\langle P_b, P_t \rangle$ a splitting of some program with nested expressions P. An interpretation I is a causal stable model of P iff there is some causal stable model J of P_b such that I is a causal stable model of $(J \cup P_t)$. Furthermore, if $\langle P_b, P_t \rangle$ is a strict splitting, then $J = I_{|S}$ where S is the set of atoms of all atoms not occurring in the head of any rule in P_t.* □

In our running example, the bottom part are $P_{7,b} = P_5$ and $P_{8,b} = P_6$ while their top part $P_{7,t} = P_{8,t}$ is the program containing the rule (13). We also can generalise this to infinite splitting sequences.

Definition 23. *A* splitting sequence *of a program P is a family $(P_\alpha)_{\alpha < \mu}$ of pairwise disjoint sets such that $P = \bigcup_{\alpha < \mu} P_\alpha$ and no atom occurring in the head of a rule in some P_α occurs in the body of a rule in $\bigcup_{\beta < \alpha} P_\beta$. A* solution *of a splitting $(P_\alpha)_{\alpha < \mu}$ is a family $(I_\alpha)_{\alpha < \mu}$ such that align=Center, leftmargin=10pt, itemindent=0.5pt*

1. *I_0 is a stable model of P_0,*
2. *I_α is a stable model of $(J_\alpha \cup P_\alpha)$ for any ordinal $0 < \alpha < \mu$ where $J_\alpha = \sum_{\beta < \alpha} I_\beta$.*

A splitting sequence is said to be strict *in α if, in addition, no atom occurring in the head of a rule in P_α occurs (the head of a rule) in $\bigcup_{\beta < \alpha} P_\beta$ and it is said to be* strict *if it is strict in α for every $\alpha < \mu$.* □

Theorem 6 (Splitting sequences)**.** *Let $(P_\alpha)_{\alpha < \mu}$ a splitting sequence of some program with nested expressions P. An interpretation I is a causal stable model of P iff there is some solution $(I_\alpha)_{\alpha < \mu}$ of $(P_\alpha)_{\alpha < \mu}$ such that $I = \sum_{\alpha < \mu} I_\alpha$. Furthermore, if such solution is strict in α, then $I_\alpha = I_{|S_\alpha}$ where S_α is the set of all atoms not occurring in the head of any rule in $\bigcup_{\alpha < \beta < \mu} P_\beta$.* □

A program P is said to be *stratified* iff there is a some ordinal μ and mapping mapping λ from the set of atoms At into the set of ordinals $\{\alpha < \mu\}$ such that, for every rule of the form (A1) and atom B occurring in the body F, it satisfies $\lambda(A) \geq \lambda(B)$ if B does not occur in the scope

of negation or a non-monotonic causal literal, and $\lambda(A) > \lambda(B)$ if B does occur under the scope of negation or a non-monotonic causal literal.

Proposition 10. *Every stratified causal program with nested expressions P has a unique causal stable model if it does not contain any rule whose head is $\bot$.* $\square$

Propositions 3, in the main part of the paper, is a direct consequence of Propositions 6 and 10.

Normal form. Proposition 6 show that Definition 22 is a conservative extension of Definitions 11. In the following we show that, in fact, the syntax of Definition 5 is a normal form, that is, for every program P in the syntax of Definition 16, there is some program Q with the syntax of Definition 5 which has exactly the same causal stable models than P.

Definition 24. *For program P and Q we write $P \Leftrightarrow Q$ when I satisfies all rules in P^J iff I satisfies all rules in Q^J for any pair of causal interpretations I and J.* $\square$

Definition 25 (Strong equivalence)**.** *Two programs P and Q are said to be* strongly equivalent *iff for every program P', $(P \cup P')$ and $(Q \cup P')$ have the same causal stable models.*

Proposition 11. *Any two causal programs P and Q s.t. $P \Leftrightarrow Q$ are strongly equivalent.* $\square$

Proposition 12. *Let P be a causal program, and let F and E be a pair of equivalent formulas, that is $F \Leftrightarrow E$. Any program obtained from P by replacing some occurrences of F by E is strongly equivalent to P.* $\square$

The following result collects some of equivalence among formulas that correspond to those in (Lifschitz et al. 1999).

Proposition 13. *For any formulas F, E and H, align=Center, leftmargin=10pt, itemindent=0.5pt*
1. *$F, E \Leftrightarrow E, F$ and $F; G \Leftrightarrow G; F$.*
2. *$F, (E, H) \Leftrightarrow (F, E), H$ and $F; (E; H) \Leftrightarrow (F; E); H$.*
3. *$F, (E; H) \Leftrightarrow (F, E); (F, H)$ and $F; (E, H) \Leftrightarrow (F; E), (F; H)$.*
4. *$\mathrm{not}(F, E) \Leftrightarrow I(\mathrm{not}\, F; \mathrm{not}\, E)$ and $\mathrm{not}(F; E)) \Leftrightarrow \mathrm{not}\, F, \mathrm{not}\, E$.*
5. *$\mathrm{not}\,\mathrm{not}\,\mathrm{not}\, F \Leftrightarrow \mathrm{not}\, F$.*
6. *$F, \top \Leftrightarrow F$ and $F; \top \Leftrightarrow \top$.*
7. *$F, \bot \Leftrightarrow \bot$ and $F; \bot \Leftrightarrow F$.*
8. *$\mathrm{not}\, \top \Leftrightarrow \bot$ and $\mathrm{not}\, \bot \Leftrightarrow \top$.* $\square$

A formula F is said to be a *simple conjunction* (resp. *simple disjunction*) iff is a conjunction (resp. disjunction) of elementary formulas.

Proposition 14. *Any formula F is equivalent to a formula of the form align=Center, leftmargin=10pt, itemindent=0.5pt*
1. *$F_1; \ldots ; F_n$ where $n \geq 1$ and each F_i is a simple conjunction, and*
2. *$F_1, \ldots, F_n$ where $n \geq 1$ and each F_i is a simple disjunction.* $\square$

Proposition 15. *A causal rule $(r_i : A \leftarrow F; E)$ is equivalent to*

$$r_i : \ A \ \leftarrow F$$
$$r_i : \ A \ \leftarrow E$$

for any label r_i, atom A and formulas F and E. $\square$

Proposition 16. *Any program is strongly equivalent of a set of rules of the form* (10) *if* $\perp$ *is allowed in the head.* ∎

Proposition 17. *For every program P, there is some program Q with the syntax of Definition 5 which has exactly the same causal stable models than P.* ∎

Appendix B.   Uniform reduct for monotonic and non-monotonic queries

An issue with Definitions 10 and 20 is that it is necessary to know whether a causal query is monotonic or not to apply the reduct. This can be provided by the user, but otherwise automatically checked whether a causal query is monotonic or not can be computationally costly. In the following, we show that, in fact, this distinction is not necessary and that the reduct can be applied uniformly to monotonic and non-monotonic causal literals.

Definition 26 (Reduct)**.** *The* reduct *of causal queries is defined as in Definition 10. The reduct of a causal literal is given by $(\psi^{I(A)} :: A)$ for any causal literal of the form of $(\psi :: A)$. The reduct of formulas, rules and programs is then defined as in Definition 20.* ∎

Definition 26 applies the reduct uniformly to monotonic and non-monotonic causal literals. A consequence of this fact is that the reduct of monotonic programs is not itself and, in fact, the least model of the reduct of a monotonic program P^I w.r.t. an interpretation I can be different according to Definitions 20 and 26. Despite that, the following result shows that the causal stable models of a program P are the same in spite of whether Definition 20 or Definition 26 is used.

Proposition 18. *Let P be a causal program with nested expressions. An interpretation I is the least model of P^I (according to Definition 20) iff I is the least model of P^I (according to Definition 26).* ∎

Appendix C.   Comparative with (Fandinno 2015b)

In this section we revise the syntax and semantics of causal programs given in (Fandinno 2015b) and show how programs in this framework can be translated in ours.

Syntax. A *m-query* is a monotonic function $\phi : \mathbf{G}_{Lb} \longrightarrow \{0,1\}$ assigning true or false to every causal graphs $G \in \mathbf{C}_{Lb}$. A *signature* is a triple $\langle At, Lb, \Phi \rangle$ where At, Lb and Φ respectively represent sets of *atoms* (or *propositions*), *labels* and query functions.

Definition 27 (m-literal)**.** *A m-literal is an expression $(\phi :: A)$ where $A \in At$ is an atom and $\phi \in \Phi$ is a m-query.* ∎

Formulas, rules and programs are defined as in our framework (Section A), but replacing causal literals (Definition 4) by m-literals (Definition 27).

Semantics. The semantics of m-programs is as follows.

Definition 28 (Valuation)**.** *The* valuation of a causal literal *of the form $(\phi :: A)$ with respect to an interpretation I is given by*

$$I(\phi :: A) \overset{\text{def}}{=} \sum \big\{\, G \in \mathbf{G}_{Lb} \;\big|\; G \leq I(A) \text{ and } \phi(G) = 1 \,\big\}$$

The valuation of causal terms and formulas is inductively defined as in Definition 17.

The definition of causal models and the T_P operator is as in Definitions 18 and 19, respectively, but evaluating formulas according to Definition 28 instead of Definition 17.

Theorem 7 (From Fandinno 2015b). *Let P be a (possibly infinite) positive logic program (with nested expressions). Then, (i) $\mathrm{lfp}(T_P)$ is the least model of P and (ii) $\mathrm{lfp}(T_P) = T_P^{\uparrow\omega}(\mathbf{0})$.* ☐

Theorem 8 (From Fandinno 2015b). *Let P be a regular positive program (with nested expressions) and Q its standard unlabelled version. Then, the least model $J = I^{cl}$ of Q is the two-valued interpretation corresponding to the least model I of P.* ☐

The definition of reduct and causal stable models is as Definitions 20 and Definition 22.

Theorem 9 (From Fandinno 2015b). *Let P be a regular program (with nested expressions) and Q be its corresponding standard program obtained by removing all labels in P. Then, P and Q are two-valued equivalent.* ☐

Encoding (Fandinno 2015b) m-programs in our framework. In the following we show that every program according to (Fandinno 2015b) can be fitted in our framework.

Definition 29. *Given a m-program Q, its corresponding program P consists of rule of the form*

$$r_i : \quad A \leftarrow F'$$

for every rule of the form $(r_i : \quad A \leftarrow F)$ in Q where F' is the result of replacing every m-query ϕ by its corresponding query ψ given by $\psi(G, t) = \phi(G)$. ☐

Proposition 19. *If P is the corresponding program of some positive m-program (with nested expressions) Q, then an interpretation I is a model of P iff I is a model of Q.* ☐

Encoding of monotonic programs into (Fandinno 2015b). It is clear that not every program in our framework can be fitted into a m-program because the last only allows monotonic queries. However, if all causal queries in a program are monotonic, then there is an equivalent m-program given as follows.

Definition 30. *Given a program with nested expressions P in which all causal queries are monotonic, its corresponding m-program Q consists of rule of the form*

$$r_i : \quad A \leftarrow F'$$

for every rule of the form $(r_i : \quad A \leftarrow F)$ in Q where F' is the result of replacing every query ψ by its corresponding query ϕ given by $\phi(G) = \psi(G, 1)$. ☐

Proposition 20. *If Q is the corresponding m-program of some positive monotonic program with nested expressions P, then an interpretation I is a model of P iff I is a model of Q.* ☐

Note that Theorem 1 is a direct consequence of Proposition 4 together with the result of Propositions 19 and 20. Furthermore, the following Corollaries 3, 4 and 5 are direct consequences of Proposition 20 together with the results of Theorems 7, 8 and 9, respectively. Corollary 6 is a direct consequence of Corollary 5.

Corollary 3. *Any (possibly infinite) positive monotonic causal program with nested expressions P has a least causal model I which (i) coincides with the least fixpoint $\mathrm{lfp}(T_P)$ of the direct consequences operator T_P and (ii) can be iteratively computed from the bottom interpretation $I = \mathrm{lfp}(T_P) = T_P^{\uparrow\omega}(\mathbf{0})$.* ☐

Corollary 4. *Let P be a regular positive monotonic program with nested expressions and Q its standard unlabelled version obtained by removing all labels from the rules in P. Let I and J be the least models of P and Q, respectively. Then, $I^{cl} = J$.* □

Corollary 5. *Let P be a regular program with nested expressions and Q be its corresponding standard program obtained by removing all labels in P. Then P and Q are two-valued equivalent.* □

Corollary 6. *Any two regular programs with nested expressions that only differ in their labels are two-valued equivalent.* □

Corollaries 1 and 2 and Theorem 2 in the main part of the paper are direct consequences of Proposition 4 plus Corollaries 3, 4 and 5, respectively.

Appendix D.   Proof of Results

Preliminary facts

Proposition 21 (From Cabalar et al. 2014a)**.** *Addition, product and application are monotonic operations, that is, $t + u \leq t' + u'$, $t * u \leq t' * u'$ and $t \cdot u \leq t' \cdot u'$ for any causal values $\{t, u, t', u'\} \subseteq \mathbf{V}_{Lb}$ such that $t \leq t'$ ant $u \leq u'$.* □

Proposition 22 (From Cabalar et al. 2014a)**.** *Every causal value $G \in \mathbf{C}_{Lb}$ without addition is completely addition-prime, that is, $G \leq \sum_{t \in T} t$ implies that $G \leq t$ for some $t \in T$ where $T \subseteq \mathbf{V}_{Lb}$ is a set of causal values.* □

Properties of the causal queries and causal literals

Proposition 23. *The evaluation of a causal literal $(\psi :: A)$ is $\leq$-monotonic for every monotonic causal query ψ, that is, $J(\psi :: A) \leq I(\psi :: A)$ for every pair of interpretations I and J such that $J \leq I$.* □

Proof. By definition, it follows that

$$X(\psi :: A) \;\overset{\text{def}}{=}\; \sum \big\{\, G \in \mathbf{C}_{Lb} \;\big|\; G \in \max X(A) \text{ and } \psi(G, X(A)) = 1 \,\big\}$$

with $X \in \{I, J\}$. For the sake of contradiction, suppose that $J(\psi :: A) \not\leq I(\psi :: A)$. Then, there is $G \in \max J(\psi :: A)$ such that $G \not\leq I(\psi :: A)$. Note that $G \in \max J(\psi :: A)$ implies $G \in \max J(A)$ and, since $J \leq I$, this implies that there exists $G' \in \max I(A)$ such that $G \leq G'$. Hence, since $J \leq I$ and ψ is monotonic, $\psi(G, J(A)) = 1$ implies $\psi(G', I(A)) = 1$ and, therefore, $G \leq G' \leq I(\psi :: A)$ which contradicts the assumption. □

Proposition 24. *The reduct of a causal query ψ w.r.t. term t, in symbols ψ^t is monotonic.* □

Proof. Suppose that ψ^t is not monotonic. Then there are $G, G'' \in \mathbf{C}_{Lb}$ and $u, w \in \mathbf{V}_{Lb}$ such that $G \leq G''$ and $\psi^t(G, u) = 1$, but $\psi^t(G'', w) = 0$. By definition,

$$\psi^t(G, u) = 1 \quad \text{iff exists some } G' \leq G \text{ s.t. } G' \in \max t \text{ and } \psi(G', t) = 1 \qquad \text{(D1)}$$

Similar for G'' and w. Pick some G satisfying (D1). Since $G' \leq G$ and $G \leq G''$, it follows that $G' \leq G''$ and, since $G' \leq G''$ and $G' \in \max t$ and $\psi(G', t) = 1$, it also follows that $\psi^t(G'', w) = \psi^t(G'', u) = 1$ which is a contradiction with the assumption. Hence, ψ^t is monotonic. □

Proposition 25. *Any monotonic causal query ψ satisfies that $\psi^t(G, u) \leq \psi(G, u)$ for any causal values $G \in \mathbf{C}_{Lb}$ and $\{t, u\} \subseteq \mathbf{V}_{Lb}$.* $\qquad\square$

Proof. Suppose that $\psi^t(G, u) \not\leq \psi(G, u)$. Then, $\psi^t(G, u) = 1$ and $\psi(G, u) = 0$. By definition,

$$\psi^t(G, u) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{iff exists some } G' \leq G \text{ s.t. } G' \in \max t \text{ and } \psi(G', t) = 1 \\ 0 & \text{otherwise} \end{cases}$$

and, thus, there exists some $G' \leq G$ such that $G' \in \max t$ and that $\psi(G', t) = 1$. Since ψ is monotonic, $G' \leq G$ and $\psi(G', t) = 1$ implies that $\psi(G, u) = 1$ for any $u \in \mathbf{V}_{Lb}$ which is a contradiction with the fact that $\psi(G, u) = 0$. $\qquad\square$

Proposition 26. *Let I and J be two interpretations. Then, $J(\psi :: A)^I \leq J(\psi :: A)$ for any atom $A \in At$ and any a monotonic causal query ψ.*

Proof. Pick any $G \in \max J(\psi :: A)^I$. By definition, $G \in \max J(A)$ and $\psi^{I(A)}(G, J(A)) = 1$. Furthermore, from Proposition 25, $\psi^{I(A)}(G, J(A)) = 1$ implies $\psi(G, J(A)) = 1$ and, thus, $G \leq J(\psi :: A)$. Therefore, $J(\psi :: A)^I \leq J(\psi :: A)$. $\qquad\square$

Note that in general $J(\psi :: A)^I \neq J(\psi :: A)$ may hold even if $J \leq I$. Consider, for instance, a pair of interpretation $J(A) = a * b$ and $I(A) = a$ and a monotonic causal query $\psi(a * b) = \psi(a) = 1$. Then, $J(\psi :: A) = a * b$, but $J(\psi :: A)^I = 0$ because $a * b \not\in \max I(A)$.

Properties of formulas

Proposition 27. *Any monotonic formula F is $\leq$-monotonic, that is, $J(F) \leq I(F)$ for any causal interpretations I and J such that $J \leq I$.* $\qquad\square$

Proof. In case that F is a causal literal of the form $(\psi :: A)$, from Proposition 23, it follows that $J(\psi :: A) \leq I(\psi :: A)$. Otherwise, we proceed by structural induction assuming the lemma holds for every subformula of F. In case that $F = (E, A)$, by induction hypothesis E and A are $\leq$-monotonic and, thus, since product is also monotonic, it follows that F is $\leq$-monotonic. The case $F = (E; A)$ is analogous. Finally, for the case $F = not\ E$, just note that F is not positive and, thus, F is not monotonic by definition. $\qquad\square$

Proposition 28. *Any causal interpretation I and formula F satisfy $I(F^I) = I(F)$.* $\qquad\square$

Proof. In case that F is a causal literal of the form $(\psi :: A)$, its reduct $(\psi :: A)^I$ is $(\psi^{I(A)} :: A)$. Furthermore, by definition,

$$\psi^{I(A)}(G, t) = 1 \quad \text{iff exist } G' \leq G \text{ s.t. } G' \in \max I(A) \text{ and } \psi(G', I(A))$$

Then, $G \in \max I(\psi^{I(A)} :: A)$ implies that $G \in \max I(A)$ and there exists some $G' \leq G$ such that $G' \in \max I(A)$ and $\psi(G', I(A)) = 1$. Note that, since $G' \leq G \in \max I(A)$ and $G' \in \max I(A)$, it follows that $G = G'$. Then, $\psi(G', I(A)) = 1$ implies that $\psi(G, I(A)) = 1$ and, consequently, $G \leq I(\psi :: A)$. That is, $I(F^I) \leq I(F)$. The other way around. $G \in \max I(\psi :: A)$ implies that $G \in \max I(A)$ and $\psi(G, I(A)) = 1$ which in turn imply that $\psi^{I(A)}(G, I(A)) = 1$ and $G \in \max I(\psi^{I(A)} :: A)$. Consequently, $I(\psi^{I(A)} :: A) = I(\psi :: A)$.

In any other case, we proceed by structural induction assuming the lemma holds for every subformula of F. In case that $F = (E, H)$, by definition,

$$I(F^I) \;=\; I(E, H)^I \;=\; I(E^I, H^I) \;=\; I(E^I) * I(H^I)$$

Furthermore, by induction hypothesis $I(E^I) = I(E)$ and $I(H^I) = I(H)$ and, thus,

$$I(F^I) \;=\; I(E) * I(H) \;=\; I(E, H) \;=\; I(F)$$

The case $F = (E; H)$ is analogous. Finally, for the case $H = not\,E$, just note that $I(not\,E)^I = I(\bot) = 0$ iff $I \models E^I$ iff $I \models E$ iff $I(not\,E) = 0$. Otherwise $I(not\,E)^I = I(\top) = 1$ and $I(not\,E) = 1$. □

Lemma D.1. *Let I and J be two interpretations. Then, $J(F^I) \leq J(F') \leq J(F)$ for any monotonic formula F and F' where F' is either F^I or the result of replacing in F^I some reduced causal query ψ^t by its non-reduced form ψ.* □

Proof. In case that F is a causal literal of the form $(\psi' :: A)$, from Proposition 26, it follows that $J(F^I) = J(\psi^{I(A)} :: A) \leq J(\psi :: A) = J(F)$. Furthermore, if in addition $\psi' = \psi$, it follows that $F' = F$ and the lemma statement follow from the above inequality. Otherwise $F' = F^I$ and the result follow in a similar way.

We proceed by structural induction assuming the statement holds for every subformula of F. In case that $F = (E, H)$, by definition,

$$J(F^I) \;=\; J((E, H)^I) \;=\; J(E^I, H^I) \;=\; J(E^I) * J(H^I)$$
$$J(F') \;=\; J((E, H)') \;=\; J(E', H') \;=\; J(E') * J(H')$$

Furthermore, by induction hypothesis, $J(E^I) \leq J(E') \leq J(E)$ and $J(H^I) \leq J(H') \leq J(H)$ and, since product $*$ is monotonic, it follows that,

$$J(F^I) \;=\; J(E^I) * J(H^I) \;\leq\; J(E') * J(H') \;=\; J(E', H') \;=\; J(F')$$
$$J(F') \;=\; J(E') * J(H') \;\leq\; J(E) * J(H) \;=\; J(E, H) \;=\; J(F)$$

The case $F = (E; H)$ is analogous. Finally, note that $F = not\,E$ is not a positive formula and, by definition, it is not a monotonic formula either. □

Proposition 29. *Let F be a monotonic formula and I be an interpretation. Then, $J(F^I) \leq J(F)$ for any interpretation J such that $J \leq I$.* □

Proof. It follows directly from Lemma D.1. □

Proposition 30. *Let F be a normal formula and I be an interpretation. Then, $J(F^I) \leq J(F)$ for any interpretation J such that $J \leq I$.* □

Proof. In case that F is a causal literal of the form $(\psi :: A)$, its reduct $(\psi :: A)^I$ is $(\psi^{I(A)} :: A)$. Note that $G \in \max J(\psi^{I(A)} :: A)$ implies $G \in \max J(A)$ and $\psi^{I(A)}(G, J(A)) = 1$. Furthermore, by definition,

$$\psi^{I(A)}(G, J(A)) = 1 \quad \text{iff exist } G' \leq G \text{ s.t. } G' \in \max I(A) \text{ and } \psi(G', I(A))$$

Since $G' \leq G \leq J(A) \leq I(A)$ and $G \in \max I(A)$, it follows that $G = G'$. Then, since $J \leq I$, queries are anti-monotonic in the second argument and $\psi(G, I(A)) = 1$, it follows that $\psi(G, J(A)) = 1$ and, since $G \in \max J(A)$, it also follows that $G \leq J(\psi :: A)$. That is, $J(F^I) \leq J(F)$

Otherwise, we proceed by structural induction assuming the lemma holds for every subformula of F. In case that $F = (E, A)$, by definition,

$$J(F^I) \;=\; J((E, H)^I) \;=\; J(E^I, H^I) \;=\; J(E^I) * J(H^I)$$

Furthermore, by induction hypothesis, $J(E^I) \leq J(E)$ and $J(H^I) \leq J(H)$ and, since product $*$ is monotonic, it follows that,

$$J(F^I) \ = \ J(E^I) * J(H^I) \ \leq \ J(E) * J(H) \ = \ J(E, H) \ = \ J(F)$$

The case $F = (E; H)$ is analogous. Finally, in case $F = not\,E$, since F is a normal formula, E is positive and every query ψ occurring in E is monotonic. Hence, from the fact $J \leq I$ and the fact that monotonic formulas are also $\leq$-monotonic, it follows $J(E) \leq I(E)$ (Proposition 27). Furthermore,

$$\text{if } J(\,(not\,E)^I\,) = 1 \text{ then } I \not\models E^I$$
$$\text{then } I(E^I) = 0$$
$$\text{then } I(E) = 0 \qquad\qquad \text{(Proposition 28)}$$
$$\text{then } J(E) = 0$$
$$\text{then } J(not\,E) = 1$$

That is, $J(\,(not\,E)^I\,) = 1$ implies that $J(not\,E) = 1$. Otherwise, $J(not\,E)^I = 0$ and the term 0 is smaller than any causal value and, thus, $J(\,(not\,E)^I\,) \leq J(not\,E)$ holds and, consequently, it follows that $J(F^I) \leq J(F)$. $\qquad\square$

Proof of Proposition 4

Proof of Proposition 4. Assume that I is a model of P w.r.t. Definition 7 and suppose that I is not a model of P w.r.t. Definition 18. Then, there is a rule r of the form of $(r_1 : A \leftarrow B_1, \ldots, B_m)$ such that $I(B_1, \ldots, B_m) \cdot r_i \not\leq A$, but that $I(B_1) * \ldots * I(B_m) \cdot r_i \leq A$. If $m > 0$, then from Definition 17 it follows that $I(B_1, \ldots, B_m) = I(B_1) * \ldots * I(B_m)$ which is a contradiction. If $m = 0$, then $\prod \emptyset = 1 = I(\top)$ which is also a contradiction. The other way around is symmetrical.

Proof of Proposition 5

Proof of Proposition 5. For (i), note that $\top = 1$, then

$$\begin{aligned}
I((F, \top)^J) \ &= \ I(F^J, \top^J) \\
&= \ I(F^J) * I(\top) \\
&= \ I(F^J) * I(1) \\
&= \ I(F^J) * 1 \\
&= \ I(F^J)
\end{aligned}$$

The remaining cases are analogous. Just note that $\bot = 0$.

Proof of Proposition 6

Proof of Proposition 6. Let r be a rule of the form of

$$r_i : \ A \leftarrow B_1, \ldots, B_m, B_{m+1}, \ldots, B_n$$

where B_j is a positive literal with $1 \leq j \leq m$ and B_j is either a negative or a consistent literal with $m + 1 \leq j \leq n$. According to Definition 20, if $I \models B_j$ with $m + 1 \leq j \leq n$, then the reduct of rule r is a rule r^I of the form

$$r_i : \quad A \leftarrow C_1, \ldots, C_m, \top, \ldots, \top$$

where C_j is the reduct of causal literal B_j for $1 \leq j \leq m$. After applying the simplifications in Proposition 5, it follows that r^I becomes

$$r_i : \quad A \leftarrow C_1, \ldots, C_m$$

which agrees with Definition 10. On the other hand, if $I \not\models B_j$ for some $m + 1 \leq j \leq n$, it follows that $B_j^I = \bot$ and, therefore,

$$(B_1, \ldots, B_m, B_{m+1}, \ldots, B_{j-1}, B_j, B_{j+1} \ldots, B_n)^I \quad = \quad \bot$$

Hence, r^I is of the from

$$r_i : \quad A \leftarrow \bot$$

and r^I does not belong to P^I after removing all rules whose body is $\bot$. Therefore, the reduct according to Definition 20 is the same as the Definition 10 for programs with the syntax of Definition 5 and the causal stable models w.r.t. Definitions 11 and 22 are the same, too.

Proof of Proposition 19

Lemma D.2. *Let F be some m-formula and F' be is corresponding formula obtained by replacing every m-query ϕ by its corresponding query ψ given by $\psi(G, t) = \phi(G)$. Then, it holds that $I(F) = I(F')$ for every interpretation I.* □

Proof. In case that $F = (\phi :: A)$ is a m-literal, by definition

$$I(\phi :: A) \quad = \quad \sum \left\{ G \in \mathbf{G}_{Lb} \mid G \leq I(A) \text{ and } \phi(G) = 1 \right\}$$

Furthermore, since ϕ is monotonic, for every $G \leq I(A)$ such that $\phi(G) = 1$, there is some G' such that $G \leq G' \in \max I(A)$ and $\phi(G) = 1$ and, thus,

$$I(\phi :: A) \quad = \quad \sum \left\{ G \in \mathbf{G}_{Lb} \mid G \in \max I(A) \text{ and } \phi(G) = 1 \right\}$$

Then, since $\psi(G, t) = \phi(G)$ for any $t \in \mathbf{V}_{Lb}$, it is clear that

$$\begin{aligned} I(\phi :: A) \quad &= \quad \sum \left\{ G \in \mathbf{G}_{Lb} \mid G \in \max I(A) \text{ and } \psi(G, I(A)) = 1 \right\} \\ &= \quad I(\psi :: A) \end{aligned}$$

In case that F is not a m-literal, the proof follows by structural induction assuming as induction hypothesis that $I(E) = I(E')$ for every subformula E of F. □

Proof of Proposition 19. Assume that I is a model and Q and suppose that I is not a model of P. Then, there is some rule r of the form $(r_i : A \leftarrow F)$ in P such that $I(F) \cdot r_1 \not\leq I(A)$. However, since r is in P there is a rule r' of the form $(r_i : A \leftarrow F)$ in Q where F' is the result of replacing every m-query ϕ by its corresponding query ψ. Then, from Lemma D.2, it follows that $I(F') = I(F)$ and, thus, $I(F') \cdot r_1 \not\leq I(A)$ which is a contradiction with the assumption

that I is a model of Q.

The other way around is symmetrical. Assume that I is a model and P and suppose that I is not a model of Q. Then, there is some rule r' of the form $(r_i : \ A \leftarrow F')$ in Q such that $I(F') \cdot r_1 \not\leq I(A)$. However, since r' is in Q there is a rule r of the form $(r_i : \ A \leftarrow F)$ in P where F' is the result of replacing every m-query ϕ by its corresponding query ψ. Then, from Lemma D.2, it follows that $I(F') = I(F)$ and, thus, $I(F) \cdot r_1 \not\leq I(A)$ which is a contradiction with the assumption that I is a model of P.

Proof of Proposition 20

Lemma D.3. *Let F be some formula and F' be is corresponding m-formula obtained by replacing every query ψ by its corresponding m-query ϕ given by $\phi(G) = \psi(G, 1)$. Then, it holds that $I(F) = I(F')$ for every interpretation I.* ☐

Proof. In case that $F = (\psi :: A)$ is a causal literal, by definition

$$I(\psi :: A) \ = \ \sum \big\{\, G \in \mathbf{G}_{Lb} \ \big| \ G \in \max I(A) \text{ and } \psi(G, I(A)) = 1 \,\big\}$$

Furthermore, since ψ is monotonic, $\psi(G, I(A)) = \psi(G, 1))$ and, thus

$$I(\psi :: A) \ = \ \sum \big\{\, G \in \mathbf{G}_{Lb} \ \big| \ G \in \max I(A) \text{ and } \psi(G, 1) = 1 \,\big\}$$

Then, since $\phi(G) = \psi(G, 1)$, it is clear that

$$
\begin{aligned}
I(\psi :: A) \ &= \ \sum \big\{\, G \in \mathbf{G}_{Lb} \ \big| \ G \in \max I(A) \text{ and } \phi(G) = 1 \,\big\} \\
&= \ I(\phi :: A)
\end{aligned}
$$

In case that F is not a m-literal, the proof follows by structural induction assuming as induction hypothesis that $I(E) = I(E')$ for every subformula E of F. ☐

Proof of Proposition 20. Assume that I is a model and Q and suppose that I is not a model of P. Then, there is some rule r of the form $(r_i : \ A \leftarrow F)$ in P such that $I(F) \cdot r_1 \not\leq I(A)$. However, since r is in P there is a rule r' of the form $(r_i : \ A \leftarrow F)$ in Q where F' is the result of replacing every m-query ϕ by its corresponding query ψ. Then, from Lemma D.3, it follows that $I(F') = I(F)$ and, thus, $I(F') \cdot r_1 \not\leq I(A)$ which is a contradiction with the assumption that I is a model of Q.

The other way around is symmetrical. Assume that I is a model and P and suppose that I is not a model of Q. Then, there is some rule r' of the form $(r_i : \ A \leftarrow F')$ in Q such that $I(F') \cdot r_1 \not\leq I(A)$. However, since r' is in Q there is a rule r of the form $(r_i : \ A \leftarrow F)$ in P where F' is the result of replacing every m-query ϕ by its corresponding query ψ. Then, from Lemma D.3, it follows that $I(F') = I(F)$ and, thus, $I(F) \cdot r_1 \not\leq I(A)$ which is a contradiction with the assumption that I is a model of P.

Proof of Theorem 1

Proof of Theorem 1. If P is the corresponding program of some positive m-program Q, the

result directly follows from Proposition 19 plus Proposition 4 and if Q is the corresponding m-program of some monotonic program P, the result directly follows from Proposition 20 plus Proposition 4.

Proof of Corollary 1 and 3

Proof of Corollary 3. This is an immediately consequence of Theorem 7 and Proposition 20. Just note that the that from Proposition 20 we can translate a program into its corresponding m-program and, then, use the T_P operator for m-programs to compute its least model. Note also that, from Lemma D.3, the T_P operators for m-programs and programs give the same results.

Proof of Corollary 1. Note that, from Proposition 4, the causal stable models of programs w.r.t. Definition 7 and 18 agree and, therefore, the statement directly follows from Corollary 3.

Proof of Corollary 2 and 4

Proof of Corollary 4. This is an immediately consequence of Theorem 8 and Proposition 20. Just note that regular programs only contain the query ψ^1 which is monotonic.

Proof of Corollary 2. Note that, from Proposition 4, the causal stable models of programs w.r.t. Definition 7 and 18 agree and, therefore, the statement directly follows from Corollary 4.

Proof of Theorem 2 and Corollary 5

Proof of Corollary 5. Suppose there is some causal stable model I of P which is not a causal stable model of Q and let P' and Q' be the corresponding m-programs of P^I and Q^I, respectively. Then, I is the least model of P^I and, from Proposition 20, I is also the least model of P^I. Just note that regular programs only contain the query ψ^1 which is monotonic. Since Q is the result of removing all labels in P, then Q^I and Q' are the result of removing all labels in P^I and P', respectively. From, Theorem 9, this implies that I is the least model of Q'. Then, from Proposition 20 again, this implies that I is the least model of Q^I which is a contradiction with the assumption that I is not a causal stable model of Q. The other way around is analogous.

Proof of Theorem 2. Note that, from Proposition 4, the causal stable models of programs w.r.t. Definition 7 and 18 agree and, therefore, the statement directly follows from Corollary 5.

Proof of Corollary 6

Proof of Corollary 6. Just note that any two programs that only differ in their labels share the same unlabelled version Q and, thus, the proof immediately follows from Corollary 2.

Proof of Proposition 18

For any program P and interpretations I and J, by $T_{P,I}(J)$ we denote an interpretation satisfying

$$T_{P,I}(J)(A) \stackrel{\text{def}}{=} \sum \{\ G \in \mathbf{C}_{Lb} \ \mid \ G \leq T_P(J)(A) \text{ and } G \in \max I(A)\ \}$$

for every atom $A \in At$.

Lemma D.4. *Let I be the least model of some monotonic program P. Then $I = T_{P,I}^{\uparrow\omega}(\mathbf{0})$.* ☐

Proof. It is clear that $T_{P,I}^{\uparrow\alpha}(\mathbf{0}) \leq T_P^{\uparrow\alpha}(\mathbf{0})$ for every ordinal α. Furthermore, from Theorem 1, it follows that $I = T_P^{\uparrow\omega}(\mathbf{0})$ and, thus, $T_{P,I}^{\uparrow\omega}(\mathbf{0}) \leq I$. Suppose for the sake of contradiction that this inequality is strict, that is, $T_{P,I}^{\uparrow\omega}(\mathbf{0}) < I$ holds. Then, there is some atom A and causal value $G \in \max I(A)$ such that $G \not\leq T_{P,I}^{\uparrow\alpha}(\mathbf{0})$ for every $\alpha < \omega$. Since $I = T_P^{\uparrow\omega}(\mathbf{0})$ and $G \leq I(A)$, it follows that there is some $\alpha < \omega$ such that $G \leq T_P^{\uparrow\alpha}(\mathbf{0})(A)$. But $G \leq T_P^{\uparrow\alpha}(\mathbf{0})(A)$ and $G \in \max I(A)$ implies that $G \leq T_{P,I}^{\uparrow\alpha}(\mathbf{0})(A)$ which is a contradiction. ☐

Lemma D.5. *Let I be the least model of some monotonic program P and α be an ordinal. Let ψ be a causal query and let Q be either P^I or the result of replacing in P^I the reduced causal query ψ^t by its non-reduced form ψ. If $T_{P,I}^{\uparrow\alpha}(\mathbf{0}) \leq T_Q^{\uparrow\alpha}(\mathbf{0}) \leq I$, then $T_{P,I}^{\uparrow\alpha}(\mathbf{0})(F) \leq T_Q^{\uparrow\alpha}(\mathbf{0})(F')$ for every monotonic formula F and F' where F' is either F^I or the result of replacing in F^I the reduced causal query ψ^t by its non-reduced form ψ.* ☐

Proof. If $F = (\psi' :: A)$ is a causal literal and $\psi' = \psi$, then $F' = F$ and the result trivially holds. Then, assume that $\psi' \neq \psi$. Thus, $G \in \max T_{P,I}^{\uparrow\alpha}(\mathbf{0})(\psi :: A)$ holds only if

- $G \in \max T_{P,I}^{\uparrow\alpha}(\mathbf{0})(A)$, and
- $\psi(G, T_{P,I}^{\uparrow\alpha}(\mathbf{0})(A)) = 1$.

By definition, it follows that $G \in \max T_{P,I}^{\uparrow\alpha}(\mathbf{0})(A)$ holds only if $G \in \max I(A)$. Furthermore, by hypothesis, it follow that $G \in \max T_{P,I}^{\uparrow\alpha}(\mathbf{0})(A) \leq T_Q^{\uparrow\alpha}(\mathbf{0})(A) \leq I(A)$. Then, $G \in \max I(A)$ and $G \leq T_Q^{\uparrow\alpha}(\mathbf{0})(A) \leq I(A)$ imply

$$G \quad \in \max \quad T_Q^{\uparrow\alpha}(\mathbf{0})(A) \tag{D2}$$

On the other hand, $G \in \max T_{P,I}^{\uparrow\alpha}(\mathbf{0})(\psi :: A)$ imply that $\psi(G, T_{P,I}^{\uparrow\alpha}(\mathbf{0})(A)) = 1$ which, since ψ is monotonic, implies that $\psi(G, I(A)) = 1$. Then, since $G \in \max I(A)$ and $\psi(G, I(A)) = 1$, it follows that $\psi^{I(A)}(G, u) = 1$ for every $u \in \mathbf{V}_{Lb}$. This plus (D2) imply $G \leq T_Q^{\uparrow\alpha}(\mathbf{0})(\psi^{I(A)} :: A)$.

Let us define the rank of a formula such that the rank of a causal literal is 0 and the rank of any other formula is the greater than the rank of all their subformulas and assume as induction hypothesis that $T_{P,I}^{\uparrow\alpha}(\mathbf{0})(E) \leq T_Q^{\uparrow\alpha}(\mathbf{0})(E')$ for every monotonic formula E of less rank than F.

In case that $F = (E, H)$, it follows that $G \in \max T_{P,I}^{\uparrow\alpha}(\mathbf{0})(F)$ holds only if there are causal values G_1 and G_2 such that $G_1 \leq T_{P,I}^{\uparrow\alpha}(\mathbf{0})(E)$ and $G_2 \leq T_{P,I}^{\uparrow\alpha}(\mathbf{0})(H)$ such that $G \leq G_1 * G_2$. Since E and H have less rank than F, by induction hypothesis, it follows that

$$G_1 \quad \leq \quad T_{P,I}^{\uparrow\alpha}(\mathbf{0})(E) \quad \leq \quad T_Q^{\uparrow\alpha}(\mathbf{0})(E') \tag{D3}$$

$$G_2 \quad \leq \quad T_{P,I}^{\uparrow\alpha}(\mathbf{0})(H) \quad \leq \quad T_Q^{\uparrow\alpha}(\mathbf{0})(H') \tag{D4}$$

and, thus, $G \leq G_1 * G_2 \leq T_Q^{\uparrow\alpha}(\mathbf{0})(F')$.

Finally, note that the case in which $F = (E; H)$ is analogous and that since F is monotonic the case $F = \textit{not } E$ is not valid. ☐

Lemma D.6. *Let I be the least model of some monotonic program P. Then, I is the least model of program Q where Q is the result of replacing some causal literal $(\psi :: A)$ in Q by its reduced form $(\psi^I(A) :: A)$.* ☐

Proof. Suppose for the sake of contradiction that I is not a model of program Q. Then, there is a rule $r' = (r_i : A \leftarrow F')$ is Q such that $I(F') \cdot r_i \not\leq I(A)$ where F' is the result of replacing in F some causal literal $(\psi :: A)$ in Q by its reduced form $(\psi^I(A) :: A)$. Since, from Lemma D.1, it follows that $I(F') \leq I(F)$ and '·' is monotonic, $I(F') \cdot r_i \not\leq I(A)$ implies that $I(F) \cdot r_i \not\leq I(A)$ which is a contradiction with the fact that I is a model of P because there is a rule $r = (r_i : A \leftarrow F)$ in P.

To show that I is the least model of Q assume as induction hypothesis that $T_{P,I}^{\uparrow\beta}(\mathbf{0}) \leq T_Q^{\uparrow\beta}(\mathbf{0})$ for every ordinal $\beta < \alpha$. Note that, if $\alpha = 0$, then $T_{P,I}^{\uparrow 0}(\mathbf{0}) = \mathbf{0}$ and, thus, the hypothesis trivially holds.

In case that α is a successor ordinal, $G \in \max T_{P,I}^{\uparrow\alpha}(\mathbf{0})(A)$ holds only if $G \in \max I(A)$ and there is some rule $r = (r_i : A \leftarrow F)$ in P and causal value $G' \in \mathbf{C}_{Lb}$ such that $G' \leq T_{P,I}^{\uparrow\alpha-1}(\mathbf{0})(F)$ and $G \leq G' \cdot r_i$. Furthermore, by induction hypothesis, it follows that $T_{P,I}^{\uparrow\alpha-1}(\mathbf{0}) \leq T_Q^{\uparrow\alpha-1}(\mathbf{0})$ and, thus, Lemma D.5 implies that $T_{P,I}^{\uparrow\alpha-1}(\mathbf{0})(F) \leq T_Q^{\uparrow\alpha-1}(\mathbf{0})(F')$ for every monotonic formula F and, thus, $G \leq T_Q^{\uparrow\alpha}(\mathbf{0})(A)$.

In case that α is a limit ordinal, $G \leq T_{P,I}^{\uparrow\alpha}(\mathbf{0})$ implies $G \leq T_{P,I}^{\uparrow\beta}(\mathbf{0})$ for some $\beta < \alpha$ which, by induction hypothesis, implies $G \leq T_Q^{\uparrow\beta}(\mathbf{0}) \leq T_Q^{\uparrow\alpha}(\mathbf{0})$.

Consequently, $T_{P,I}^{\uparrow\alpha}(\mathbf{0}) \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})$ for every ordinal α. Furthermore, from Theorem 1, it follows that $T_Q^{\uparrow\omega}(\mathbf{0})$ is the least model of Q and, from Lemma D.4 and the fact that I is the least model of P it follows that $I = T_{P,I}^{\uparrow\omega}(\mathbf{0})$. Since I is a a model of Q and $I \leq T_Q^{\uparrow\omega}(\mathbf{0})$, it follows that I must be the least model of Q. $\square$

Proof of Proposition 18. Let Q be the reduct of program P w.r.t. I and Definition 10 and Q' be the reduct of program P w.r.t. I and Definition 26. Then, Q is monotonic and, from Lemma D.6, it follows that I is the least model of Q iff I is the least model of Q'.

Proof of Proposition 7

Proof of Proposition 7. Suppose that I is not a model of P. Then there is a rule r in P of the form of (10) such that $I(F) \cdot r_i \not\leq I(A)$. Since rule r is in P, rule r^I of the form

$$r_i : \quad A \leftarrow F^I \tag{D5}$$

is in P^I. Furthermore, $I(F) = I(F^I)$ from Proposition 28 and, thus, $I(F^I) \cdot r_i \not\leq I(A)$. That is, I is not a model of r^I and, consequently, is not a model of P^I which contradicts the assumption that I is a causal stable model of P.

Lemma D.7. *Let P be a program, I be an interpretation and α be an ordinal. Let Q be the result of replacing in P^I the reduced causal query ψ^t of every monotonic query by its non-reduced form ψ. If $T_{Q,I}^{\uparrow\alpha}(\mathbf{0}) \leq T_{PI}^{\uparrow\alpha}(\mathbf{0}) \leq I$, then $T_{Q,I}^{\uparrow\alpha}(\mathbf{0})(F') \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(F^I)$ for every monotonic formulas F' and F^I where F' is the result of replacing in F^I the reduced causal query ψ^t of every monotonic query by its non-reduced form ψ.* $\square$

Proof. If $F = (\psi :: A)$ is a causal literal and ψ is not a monotonic causal query, then $F' = F^I$

and the result trivially holds. Then, assume that ψ is a monotonic causal query. This implies that $G \in \max T_{Q,I}^{\uparrow\alpha}(\mathbf{0})(\psi :: A)$ holds only if

- $G \in \max T_{Q,I}^{\uparrow\alpha}(\mathbf{0})(A)$, and
- $\psi(G, T_{Q,I}^{\uparrow\alpha}(\mathbf{0})(A)) = 1$.

By definition, it follows that $G \in \max T_{Q,I}^{\uparrow\alpha}(\mathbf{0})(A)$ holds only if $G \in \max I(A)$. Furthermore, by hypothesis, it follow that $G \in \max T_{Q,I}^{\uparrow\alpha}(\mathbf{0})(A) \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(A) \leq I(A)$. Then, $G \in \max I(A)$ and $G \leq T_{Q,I}^{\uparrow\alpha}(\mathbf{0})(A) \leq I(A)$ imply

$$G \in \max \ T_{PI}^{\uparrow\alpha}(\mathbf{0})(A) \tag{D6}$$

On the other hand, $G \in \max T_Q^{\uparrow\alpha}(\mathbf{0})(\psi :: A)$ imply that $\psi(G, T_Q^{\uparrow\alpha}(\mathbf{0})(A)) = 1$ which, since ψ is monotonic, implies that $\psi(G, I(A)) = 1$. Then, since $G \in \max I(A)$ and $\psi(G, I(A)) = 1$, it follows that $\psi^{I(A)}(G, u) = 1$ for every causal value $u \in \mathbf{V}_{Lb}$. This plus (D6) imply that $G \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(\psi^{I(A)} :: A) = T_{PI}^{\uparrow\alpha}(\mathbf{0})(F^I)$.

Let us define the rank of a formula such that the rank of a causal literal is 0 and the rank of any other formula is the greater than the rank of all their subformulas and assume as induction hypothesis that $T_{Q,I}^{\uparrow\alpha}(\mathbf{0})(E') \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(E^I)$ for every monotonic formula E of less rank than F.

In case that $F = (E, H)$, it follows that $G \in \max T_Q^{\uparrow\alpha}(\mathbf{0})(F)$ holds only if there are causal values G_1 and G_2 such that $G_1 \leq T_Q^{\uparrow\alpha}(\mathbf{0})(E')$ and $G_2 \leq T_Q^{\uparrow\alpha}(\mathbf{0})(H')$ such that $G \leq G_1 * G_2$. Since E and H have less rank than F, by induction hypothesis, it follows that

$$G_1 \ \leq \ T_Q^{\uparrow\alpha}(\mathbf{0})(E') \ \leq \ T_Q^{\uparrow\alpha}(\mathbf{0})(E^I) \tag{D7}$$
$$G_2 \ \leq \ T_Q^{\uparrow\alpha}(\mathbf{0})(H') \ \leq \ T_Q^{\uparrow\alpha}(\mathbf{0})(H^I) \tag{D8}$$

and, thus, $G \leq G_1 * G_2 \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(F^I)$.

The case in which $F = (E; H)$ is analogous. In case that $F = not\, E$, by definition it follows that $F' = \bot$ iff $F^I = \bot$ and $F' = \top$ iff $F^I = \top$ □

Proof of Proposition 1 and 8

Lemma D.8. *The reduct F^I of a formula F w.r.t. any interpretation I is $\leq$-monotonic, that is, $J(F^I) \leq K(F^I)$ for all causal interpretations J and K such that $J \leq K$.* □

Proof. From Proposition 24, the reduct of any query $\psi^{I(A)}$ is monotonic. Furthermore, the reduct of any formula F^I is positive. Hence, F^I is monotonic and, from Proposition 27, it follows that formula F^I is $\leq$-monotonic. □

Proof of Proposition 8. From Proposition 7, any causal stable model I of a program P is a model of P. Suppose that I is not supported, that is, there is some true atom A and cause $G \leq I(A)$ sucht that no rule r in P of the form of (10) satisfies $G \leq I(F) \cdot r_i$. Furthermore, from Proposition 28, it follows that $I(F^I) = I(F)$. That is, no rule r^I in P^I satisfies $G \leq I(F^I) \cdot r_i$.

Let J be a causal interpretation such that $J(B) = I(B)$ for every atom $B \neq A$ and $J(A) = \sum\{\, G' \in \mathbf{C}_{Lb} \mid G' \leq I(A) \text{ and } G \not\leq G' \,\}$. Clearly $J < I$ and, since I is a $\leq$-minimal model of P^I, J cannot be a model of P^I. That is, there is a rule r^I in P^I of the form of (10) such that

$J(F^I) \cdot r_i \not\leq J(A)$. Then there is a cause $G' \leq J(F^I) \cdot r_i$ such that $G' \not\leq J(A)$. Since $I \leq J$, it follows that $J(F^I) \leq I(F^I)$ (Lemma D.8) and thus, since application is monotonic, it follows that $G' \leq I(F^I) \cdot r_i$. Note that $G' \leq I(F^I) \cdot r_i$, but no rule in P^I with A in the head satisfies $G \leq I(F^I) \cdot r_i$. Then $G \not\leq G'$. Moreover, since $I \models r^I$, it follows that $G' \leq I(A)$ and then, since $G \not\leq G'$, it follows that $G' \leq J(A)$, which is a contradiction with the fact that $G' \not\leq J(A)$. Consequently, I is a supported model of P.

Proof of Proposition 1. Note that, from Proposition 4, the causal stable models of programs w.r.t. Definition 7 and 18 agree and, therefore, the statement directly follows from Proposition 8.

Proof of Proposition 2 and 9

Lemma D.9. *Let P be a normal program and I and J be two causal interpretation such that $J \leq I$. If J is a model of P, then J is a model of P^I.* ☐

Proof. Suppose that J is a model of P and not a model of P^I. Then, there is a rule r in P of the form of (10) such that $J(F) \cdot r_i \leq J(A)$ and $J(F^I) \cdot r_i \not\leq J(A)$. Note that, since P is a normal program, the formula F must also be normal. Then, since $J \leq I$, Proposition 30 implies that $J(F^I) \leq J(F)$. Furthermore, since application '·' is monotonic, it follows that

$$J(F^I) \cdot r_i \ \leq \ J(F) \cdot r_i \ \leq \ J(A)$$

which is a contradiction with the fact that $J(F^I) \cdot r_i \not\leq J(A)$. ☐

Proof of Proposition 9. If I is a causal stable model of P, then, Proposition 7 implies that I is a model of P. Suppose that I is not $\leq$-minimal. Then there exists an interpretation $J \leq I$ such that J is a model of P. But, since P is a normal program, from Lemma D.9, J must be a model of P^I and, thus, I is not a $\leq$-minimal model of P^I which contradicts the assumption that I is a causal stable model of P.

Proof of Proposition 2. Note that, from Proposition 4, the causal stable models of programs w.r.t. Definition 7 and 18 agree and, therefore, the statement directly follows from Proposition 9.

Proof of Theorem 3 and 5

Definition 31. *A* splitting *of a program P is a pair $\langle P_b, P_t \rangle$ of pairwise disjoint sets such that $P = (P_b \cup P_t)$ and no atom occurring in the head of a rule in P_t occurs in the body of a rule in P_b. A splitting is said to be* strict *if, in addition, no atom occurring in the head of a rule in P_t occurs (the head of a rule) in P_b.* ☐

Lemma D.10. *Let P_b and P_t be two monotonic programs such that no atom occurring in a body in P_b is a head atom of P_t. Let I and J be the least models of $(P_b \cup P_t)$ and P_b, respectively. Then, I is also the least model of program $(J \cup P_t)$. Furthermore, $J_{|S} = I_{|S}$ where S is the set of atoms of all atoms not in the head of any rule in P_t.* ☐

Proof. Since interpretation J is the least model of the program J and $J \leq I$, it follows that I satisfies all rules in program J. In addition, since I is the least model of program $(P_b \cup P_t)$, it is clear that I also satisfies all rules in P_t and, thus, I satisfies all rules in program $(J \cup P_t)$. Suppose that I is not the least model of $(I \cup P_t)$. Then, there is a model I' of $(J \cup P_t)$ such

that $I' < I$. Since I is the least model of program $(P_b \cup P_t)$ and $I' < I$, it follow that I' does not satisfy some rule $r = (r_i : A \leftarrow F)$ in $(P_b \cup P_t)$. That is, $I'(F) \cdot r_i \not\leq I'(A)$. Since I' is a model of $(J \cup P_t)$, it is clear that $J \leq I'$ and, since in addition $I' < I$, it follows that $I(F) \cdot r_i \not\leq J(A)$ also holds. Furthermore, I' satisfy all rules in P_t because I' is a model of $(J \cup P_t)$ and, thus, rule r must be in P_b and no atom occurring in F occurs in the head of a rule in P_t. Hence, $I(F) = J(F)$ and, thus, $I(F) \cdot r_i \not\leq J(A)$ implies that $J(F) \cdot r_i \not\leq J(A)$ which is a contradiction with the hypothesis that J is a model of P_b and the fact that r in P_b. Consequently, I is also the least model of program $(J \cup P_t)$. Furthermore, since I is the least model of program $(J \cup P_t)$ and no atom in S occurs in the head of any rule in P_t, it follows that $I_{|S} = J_{|S}$. $\qquad\square$

Proof of Theorem 5. For the only if direction. Assume that I is a causal stable model of program $(P_b \cup P_t)$. Then, I is the least model of the monotonic program $(P_b \cup P_t)^I = (P_b^I \cup P_t^I)$. Let J be the least model of P_b^I. Since I and J respectively are the least models of $(P_b^I \cup P_t^I)$ and P_b^I and no atom occurring in a body in P_b^I is in the head of any rule in P_t^I, from Lemma D.10, it follows that I is the least model of program $(J \cup P_t^I) = (J \cup P_t)^I$ and, consequently, I is a causal stable model of $(J \cup P_t)$ and $I_{|S} = J_{|S}$ where S is the set of atoms of all atoms not occurring in the head of any rule in P_t. In addition, since $I_{|S} = J_{|S}$ and all atoms in the body of some rule in P_b are in S, it follows that $P_b^I = P_b^J$ and, therefore, J is the least model of $P_b^I = P_b^J$ and a causal stable model of P_b. Furthermore, if no atom occurring in P_b occurs in the head of a rule in P_t, then $J_{|S} = J$ (note that S contains all atoms in P_b since no atom occurring in P_b occurs in the head of a rule in P_t) and, thus, $I_{|S} = J$.

The other way around. If I is a causal stable model of $(J \cup P_t)$, then I is the least model of $(J \cup P_t)^I = (J \cup P_t^I)$. Let S be the of all atoms not occurring in the head of a rule in P_t. Then, S contains all atoms occurring in the body of the rules in P_b and, since I is the least model of $(J \cup P_t^I)$, it follows that $I_{|S} = J_{|S}$ and, thus, $P_b^I = P_b^J$. Then, since J is a causal stable model of P_b, it follows that J is the least model of P_b^I. From Lemma D.10, this implies that I is the least model of program $(P_b^I \cup P_t^I) = (P_b \cup P_t)^I = P^I$ and, thus, I is a causal stable model of P.

Proof of Theorem 3. Note that, from Proposition 4, the causal stable models of programs w.r.t. Definition 7 and 18 agree and, therefore, the statement directly follows from Theorem 5.

Proof of Theorem 6

Lemma D.11. *Let $(P_\alpha)_{\alpha < \mu}$ a splitting sequence of some monotonic program P. Then, there is a unique solution $(I_\alpha)_{\alpha < \mu}$ of $(P_\alpha)_{\alpha < \mu}$ and it satisfies (i) $I = \sum_{\alpha < \mu} I_\alpha$ and (ii) $I_{\alpha|S_\alpha} = I_{|S_\alpha}$ where I is the least model of P and S_α is the set of all atoms not occurring in the head of any rule in $\bigcup_{\alpha < \beta < \mu} P_\beta$.* $\qquad\square$

Proof. First note that, since P is a monotonic program, every P_α with $\alpha < \mu$ is also monotonic and, thus, there is a unique causal stable model I_0 of P_0. Suppose that there is a solution $(I'_\alpha)_{\alpha < \mu}$ of $(P_\alpha)_{\alpha < \mu}$ such that $I'_\alpha \neq I_\alpha$ for some $\alpha < \mu$. Let α be the first ordinal such that $I'_\alpha \neq I_\alpha$. Then, $0 < \alpha < \mu$ and there are two different causal stable models I_α and I'_α of $(J_\alpha \cup P_\alpha)$ which is a contradiction with the fact that $(J_\alpha \cup P_\alpha)$ is monotonic.

Let $I = \sum_{\alpha < \mu} I_\alpha$ and we will show that I is the least model of P and that $I_\alpha = I_{|S_\alpha}$. Assume

as induction hypothesis that the lemma statement holds for every ordinal $\mu' < \mu$ and note that, in case that $\mu = 0$, it follows that $P = \bigcup_{\alpha<0} P_\alpha = \emptyset$ and that $I = \sum_{\alpha<0} I_\alpha = \mathbf{0}$ and that $\mathbf{0}$ is the least model of the empty program.

In case that μ is a successor ordinal, let $\mu' = \mu - 1$ be its predecessor, let $Q = \bigcup_{\alpha<\mu'} P_\alpha$ and J be the least model of Q. Then, $(I_\alpha)_{\alpha<\mu'}$ is solution of $(P_\alpha)_{\alpha<\mu'}$, $\langle Q, P_{\mu'}\rangle$ is a splitting of P, and, by induction hypothesis $J = \sum_{\alpha<\mu'} I_\alpha$ and $I_{\alpha|S_\alpha} = J_{|S_\alpha}$ for every $\alpha < \mu'$.

Let $I_{\mu'}$ be the least model of $(J \cup P_{\mu'})$. Since $I_{\mu'}$ is the least model of $(J \cup P_{\mu'})$, it follows that $I_{\mu'} \geq J$ and, thus, $I = \sum_{\alpha<\mu} I_\alpha = I_{\mu'} + \sum_{\alpha<\mu'} I_\alpha = I_{\mu'} + J = I_{\mu'}$. That is, $I = I_{\mu'}$ is the least model of $(J \cup P_{\mu'})$ and, since J is the least model of Q, from Lemma D.10, it follows that I is the least model of $P = (Q \cup P_{\mu'})$ and, that, $I_{\mu'|S_{\mu'}} = I_{|S_{\mu'}}$. Furthermore, since no atom in S_α with $\alpha < \mu'$ occurs in the head of any rule in $P_{\mu'}$ it follows that $I_{|S_\alpha} = J_{|S_\alpha}$ for every $\alpha < \mu'$. Consequently $I_{\alpha|S_\alpha} = I_{|S_\alpha}$ for every $\alpha < \mu$.

In case that μ is a limit ordinal, by induction hypothesis $I_{\alpha|S_\alpha} = I_{|S_\alpha}$ for every $\alpha < \mu'$ and, thus, since all atoms occurring in the body of any rule in P_α belong to S_α, it follows that $P_\alpha^{I_\alpha} = P_\alpha^I$. Furthermore, since $(I_\alpha)_{\alpha<\mu}$ is solution of $(P_\alpha)_{\alpha<\mu}$, it follows that I_α is the least model of $(J_\alpha \cup P_\alpha)$ and, thus, I_α is a model of $P_\alpha^{I_\alpha} = P_\alpha^I$. Since $I = \sum_{\alpha<\mu} I_\alpha \geq I_\alpha$, then I is a model of P_α^I for every $\alpha < \mu'$ and, consequently, I is a model of P^I.

Suppose that I is not the least model of P. Then, there is a model I' of P such that $I' < I$. Since $I = \sum_{\alpha<\mu} I_\alpha$ and $I' < I$, it follows that $I_\alpha \not\leq I'$ for some first ordinal $\alpha < \mu$. Since α is the first ordinal such that $I_\alpha \not\leq I'$, it follows that $J_\alpha = \sum_{\beta<\alpha} I_\beta \leq I'$ and, thus, I' satisfies all rules in J_α. Furthermore, since $P_\alpha \subseteq P$ and I' is model of P, it follows that I' also satisfies all rules in P_α. That is, I' is a model of $(J_\alpha \cup P_\alpha)$ and $I_\alpha \not\leq I'$ which is a contradiction with the fact that I_α is the least model of $(J_\alpha \cup P_\alpha)$. Consequently, I is the least model of P.

Suppose now that $I_{\alpha|S_\alpha} \neq I_{|S_\alpha}$ for some $\alpha < \mu$ and let α be the first such ordinal. Then, there is some first ordinal α' and atom $A \in S_\alpha$ such that $I_\alpha(A) \not\leq I_{\alpha'}(A)$. Note that $\alpha' \leq \alpha$ implies that $I_{\alpha'} \leq I_\alpha$ and, thus, it must be that $\alpha < \alpha'$. Since α' first ordinal that satisfies $I_\alpha(A) \not\leq I_{\alpha'}(A)$ it follows that $I_\beta(A) \leq I_\alpha(A)$ for every $\beta < \alpha'$ and, thus, $J_{\alpha'}(A) \leq I_\alpha(A)$. Since $J_{\alpha'}(A) \leq I_\alpha(A) \not\leq I_{\alpha'}(A)$ and $I_{\alpha'}$ is the least model of $(J_{\alpha'} \cup P_{\alpha'})$, there must be some rule $r = (r_i : A \leftarrow F) \in P_{\alpha'}$ which is a contradiction with the fact that $A \in S_\alpha$ and $\alpha < \alpha'$. Consequently, $I_{\alpha|S_\alpha} = I_{|S_\alpha}$ for all $\alpha < \mu$. $\qquad\square$

Proof of Theorem 6. For the only if direction. Assume that I is a causal stable model of P. Then, I is the least model of the monotonic program P^I and, from Lemma D.11 there is a unique solution $(I_\alpha)_{\alpha<\mu}$ of program P^I and it satisfies (i) $I = \sum_{\alpha<\mu} I_\alpha$ and (ii) $I_{\alpha|S_\alpha} = I_{|S_\alpha}$. Furthermore, by definition, align=Center, leftmargin=10pt, itemindent=0.5pt

1. I_0 is the least model of P_0^I,
2. I_α is a stable model of $(J_\alpha \cup P_\alpha^I)$ for any ordinal $0 < \alpha < \mu$ where $J_\alpha = \sum_{\beta<\alpha} I_\beta$.

Since $I_{\alpha|S_\alpha} = I_{|S_\alpha}$ and all atoms occurring in the body of any rule in P_α belong to S_α, it follows that $P_\alpha^I = P_\alpha^{I_\alpha}$ and, thus, align=Center, leftmargin=10pt, itemindent=0.5pt

1. I_0 is the least model of $P_0^{I_\alpha}$,
2. I_α is a stable model of $(J_\alpha \cup P_\alpha)^{I_\alpha} = (J_\alpha \cup P_\alpha^{I_\alpha})$ for any ordinal $0 < \alpha < \mu$ where $J_\alpha = \sum_{\beta<\alpha} I_\beta$.

Consequently, $(I_\alpha)_{\alpha<\mu}$ is a solution of $(P_\alpha)_{\alpha<\mu}$ and it satisfies $I = \sum_{\alpha<\mu} I_\alpha$ and $I_{\alpha|S_\alpha} = I_{|S_\alpha}$.

The other way around. Assume there is some solution $(I_\alpha)_{\alpha<\mu}$ of $(P_\alpha)_{\alpha<\mu}$ and let $I = \sum_{\alpha<\mu} I_\alpha$. By definition, align=Center, leftmargin=10pt, itemindent=0.5pt

1. I_0 is the least model of $P_0^{I_0}$,
2. I_α is the least model of $(J_\alpha \cup P_\alpha^{I_\alpha})$ for any ordinal $0 < \alpha < \mu$ where $J_\alpha = \sum_{\beta<\alpha} I_\beta$.

Since S_α contains all atoms not in the head of any rule in $\bigcup_{\alpha<\beta<\mu} P_\beta$, it follows that

$$\sum_{\beta<\alpha} I_{\beta|S_\alpha} = J_{\alpha|S_\alpha} \leq I_{\alpha|S_\alpha} = J_{\alpha+1|S_\alpha} = I_{\alpha+1|S_\alpha} = \ldots = \sum_{\beta<\mu} I_{\beta|S_\alpha} = I_{|S_\alpha}$$

and, since S_α contains all atoms occurring in the body of all rules in P_α, it follows that $P_\alpha^I = P_\alpha^{I_\alpha}$ and, thus, align=Center, leftmargin=10pt, itemindent=0.5pt

1. I_0 is the least model of P_0^I,
2. I_α is the least model of $(J_\alpha \cup P_\alpha^I)$ for any ordinal $0 < \alpha < \mu$ where $J_\alpha = \sum_{\beta<\alpha} I_\beta$.

Hence, $(I_\alpha)_{\alpha<\mu}$ of $(P_\alpha^I)_{\alpha<\mu}$ and, from Lemma D.11, it follows that I is the least model of P^I and a causal stable model of P.

Furthermore, if $(I_\alpha)_{\alpha<\mu}$ is a strict solution in α, then no atom occurring in P_α occurs in the head of a rule in any P_β with $\alpha < \beta < \mu$, and, thus, every atom occurring in $(J_\alpha \cup P_\alpha)$ belongs to S_α. Consequently, $I_\alpha = I_{\alpha|S_\alpha} = I_{|S_\alpha}$.

Proof of Proposition 3 and 10

Proof of Proposition 10. Let $P_{\alpha+1}$ be the set of rules of the form of (A1) such that $\lambda(A) = \alpha$ and $P_\alpha = \emptyset$ if α is a limit ordinal. Then, $(P_\alpha)_{\alpha<\mu}$ is a strict splitting sequence of P and, from Theorem 6, an interpretation I is a causal stable model of P iff there is some solution $(I_{|S_\alpha})_{\alpha<\mu}$ of $(P_\alpha)_{\alpha<\mu}$ such that $I = \sum_{\alpha<\mu} I_{|S_\alpha}$. where S_α is the set of all atoms not occurring in the head of any rule in $\bigcup_{\alpha<\beta<\mu} P_\beta$. Hence, it is enough to show that every P_α has a unique causal stable model.

By definition, it is clear that P_α has the $\mathbf{0}$ interpretation as its unique causal stable model when α is a limit ordinal. In case that α is a successor ordinal, suppose that there are two different causal stable models J and J' of P_α. Since P is stratified, there is no rule in $\bigcup_{\alpha-1<\beta<\mu} P_\beta$ with an atom occurring in P_α under the scope of negation or a non-monotonic causal literal in P_α. Hence, $J(B) = J'(B) = I_{|S_{\alpha-1}}(B)$ for every atom B occurring under the scope of negation or a non-monotonic causal literal and, thus, $P^J = P^{J'}$ and J and J' must be equal which is a contradiction with the assumption.

Proof of Proposition 3. Note that, from Proposition 4, the causal stable models of programs w.r.t. Definition 7 and 18 agree and, therefore, the statement directly follows from Proposition 10. Just note that, according to Definition 5, $\bot$ is not allowed in the head of the rules.

Proof of Proposition 11. Let R be any causal program over the signature σ of P and Q. Let $\mathcal{I}$, $\mathcal{J}$ respectively be the sets of causal stable models of program $P \cup R$ and $Q \cup R$. Any causal stable model $I \in \mathcal{I}$ is the least model of the positive program $(P \cup R)^I = P^I \cup R^I$. That is, I satisfies all rules in both P^I and R^I and, since $P \Leftrightarrow Q$, I satisfies all rules in Q^I. Suppose there exists I' which satisfies all rules in $(Q \cup R)^I$ and $I' < I$. By the same reasoning I' satisfies all rules in P^I (an also in R^I) contradicting the assumption that I is the lest model of $(P \cup R)^I$. Hence, I is the least model of $(P \cup R)^I$, and so, an stable model of $(P \cup R)$. That is, $I \in \mathcal{J}$. The other way around is analogous.

Proof of Proposition 12

Lemma D.12. *Let F, G and H be formulas such that $F \Leftrightarrow G$. If a formula H' is obtained from H by replacing some regular occurrences of F by G, then $H \Leftrightarrow H'$.* $\quad\square$

Proof. By structural induction like Lemma 4 in (Lifschitz et al. 1999). If H is elementary. Then either $H = F$ and $H' = G$ or $H = H'$. In both cases $H \Leftrightarrow H'$. Otherwise, if $H = F$ and $H' = G$, then also $H \Leftrightarrow H'$. Hence, in the following we assume that $H \neq F$.

1. In case $H = H_1, H_2$, then $H' = H_1', H_2'$ and, by induction hypothesis, $H_i \Leftrightarrow H_i'$ with $i \in \{1, 2\}$. Then

$$
\begin{aligned}
I(H^J) &= I((H_1, H_2)^J) \\
&= I(H_1^J, H_2^J) \\
&= I(H_1^J) * I(H_2^J) \\
&= I((H_1')^J) * I((H_2')^J) \\
&= I((H_1', H_2')^J) \\
&= I((H')^J)
\end{aligned}
$$

2. The case $H = H_1; H_2$ is similar to the previous one.
3. In case $H = \textit{not } H_1$, then $H' = \textit{not } H_1'$ and, by induction hypothesis, $H_1 \Leftrightarrow H_1'$.

$$
\begin{aligned}
I(H^J) = 1 \ \ &\text{iff} \ \ I((\textit{not } H_1)^J) = 1 \\
&\text{iff} \ \ J(H_1^J) = 0 \\
&\text{iff} \ \ J((H_1')^J) = 0 \\
&\text{iff} \ \ I((\textit{not } H_1')^J) = 1 \\
&\text{iff} \ \ I((H')^J) = 1
\end{aligned}
$$

and $I(H^J) = 0$ otherwise, that is, iff $I((H')^J) = 0$ $\quad\square$

$\quad\square$

Proof of Proposition 12. Similar to the proof of Proposition 3 in (Lifschitz et al. 1999). Let Q be the program obtained by replacing some occurrences of F by G in P. Assume that I is satisfies all rules in Q^J. Take any rule $(r_i : A \leftarrow F)$ in P. Its corresponding rule $(r_i : A \leftarrow E)$ in Q must satisfy

$$
I(E^J) \cdot r_i \leq I(A)
$$

and, by Lemma D.12, it follows that $I(F^J) = I(E^J)$. Consequently,

$$I(F^J) \cdot r_i \leq I(A)$$

Hence, I satisfies all rules in P. The other way around is similar. Hence, I satisfies all rules in P^J iff I satisfies all rules in Q^J. That is $P \Leftrightarrow Q$ and, by Proposition 11, P and Q are strongly equivalent.

Proof of Proposition 13

Proof of Proposition 13. For (i) note that

$$
\begin{aligned}
I((F, E)^J) &= I(F^J, E^J) \\
&= I(F^J) * I(E^J) \\
&= I(E^J) * I(F^J) \\
&= I((E, F)^J)
\end{aligned}
$$

Similarly, $I((F; E)^J) = I((E; F)^J)$. Note that product and addition are both commutative. The same reasoning applies for (ii) and (iii) by noting that product and addition are also associative and distributes over one over the other.

For (iv),

$$
\begin{aligned}
I((not\,not\,not\,F)^J) = 1 \text{ iff } & J((not not\,F)^J) = 0 \\
& \text{iff } J((not\,F)^J) = 1 \\
& \text{iff } J((not\,F)^J) = 1 \\
& \text{iff } J(F^J) = 0 \\
& \text{iff } I((not\,F)^J) = 1
\end{aligned}
$$

and $I((not\,not\,not\,F)^J) = 0$ otherwise, that is $I((not\,F)^J) = 0$.

Similarly, for (v),

$$
\begin{aligned}
I((not(F; E))^J) = 1 \text{ iff } & J(F^J; E^J) = 0 \\
& \text{iff } J(F^J) + J(E^J) = 0 \\
& \text{iff } J(F^J) = 0 \text{ and } J(E^J) = 0 \\
& \text{iff } I((not\,F)^J) = 1 \text{ and } I((not\,E)^J) = 1 \\
& \text{iff } I((not\,F)^J) * I((not\,E)^J) = 1 \\
& \text{iff } I((not\,F, not\,E)^J) = 1
\end{aligned}
$$

and $I(not(F; E)^J) = 0$ otherwise, that is $I((not\,F, not\,E)^J) = 0$. Furthermore

$$
\begin{aligned}
I((not(F, E))^J) = 1 \text{ iff } & J((F, E)^J) = 0 \\
\text{iff } & J(F^J) * J(E^J) = 0 \\
\text{iff } & J(F^J) = 0 \text{ or } J(E^J) = 0 \\
\text{iff } & I((not\,F)^J) = 1 \text{ or } I((not\,E)^J) = 1 \\
\text{iff } & I((not\,F)^J) + I((not\,E)^J) = 1 \\
\text{iff } & I((not\,F; not\,E)^J) = 1
\end{aligned}
$$

and $I((not(F, E))^J) = 0$ otherwise, that is $I((not\,F; not\,E)^J) = 0$. (vi) and (vii) directly follows from Proposition 5. Finally, for $(viii)$, $I(not\,\top) = 0 = \bot$ and $I(not\,\bot) = 1 = \top$.

Proof of Proposition 14

Proof of Proposition 14. The proof follows by structural induction using Proposition 13 and Lemma D.12 exactly as in (Lifschitz et al. 1999). Note that we do not consider strong negation, so all formulas are regular.

Proof of Proposition 15

Proof of Proposition 15. Note that $I \models (r_i : A \leftarrow F; E)^J$ iff $I \models (r_i : A \leftarrow F^J; E^J)$

$$
\big(I(F^J) + I(E^J)\big) \cdot r_i \leq I(A)
$$

which, by application distributivity over addition, is equivalent to

$$
I(F^J) \cdot r_i + I(G^J) \cdot r_i \leq I(A)
$$

which in turn holds iff $I \models (r_i : A \leftarrow F)^J$ and $I \models (r_i : A \leftarrow E)^J$.

Proof of Proposition 16

Proof of Proposition 16. Propositions 11, 12 and 14 show that any program is strongly equivalent to a set of rules of the form

$$
r_i : A \leftarrow F_1; \ldots ; F_m, \tag{D9}
$$

where each F_i is a simple conjunction. Similarly, Propositions 11, 12 and 15 show that such set of rules is strongly equivalent to a set of rules of the form

$$
r_i : A \leftarrow F \tag{D10}
$$

where each F is a simply conjunction. That is, a set of rule of the form (10) in which the head can be $\bot$.

Proof of Proposition 17. From Proposition 17, every program can be writing as an equivalent program where all rules r are of the form

$$r_i : \ A \ \leftarrow \ B_1, \ldots, B_m \tag{D11}$$

where A is an atom or $\bot$. If A is an atom, then r is already of the form of (10). Otherwise, replace rule r by a rule r' of the form of

$$r_i : \ aux_r \ \leftarrow \ B_1, \ldots, B_m, not \, aux_r \tag{D12}$$

where aux_r is a new auxiliary predicate. Let Q be the result of replacing r by r' in P. If I is a causal stable model of P, then $I \not\models B_j$ for some $1 \leq j \leq m$ and, thus, it is a causal stable model of Q. The other way around, if I is a causal stable model of Q, either $I \models aux_r$ or $I \not\models B_j$ for some $1 \leq j \leq m$. If the former, rule r' does not belong to Q^I and, thus, there is no rule which aux_r which contradicts the fact that I must be the least model of Q^J. Hence, $I \not\models aux_r$ and $I \not\models B_j$ for some $1 \leq j \leq m$ and, therefore, I is a causal stable model of P.

Appendix E.   Complexity assessment

First, it has been showed in (Cabalar et al. 2014b) that there may an exponential number of causes for some atom with respect to a casual stable model. For instance, consider the positive program P_{16} consisting of following the rules:

$a : p_1$	$b : p_1$	$m_i : p_i \leftarrow p_{i-1}, \ q_{i-1}$	for $i \in \{2, \ldots, n\}$
$c : q_1$	$d : q_1$	$n_i \ : q_i \leftarrow p_{i-1}, \ q_{i-1}$	for $i \in \{2, \ldots, n\}$

Since program P_{16} is positive it has unique causal stable model I_{16}. Furthermore, it is easy to see that the interpretation of atoms p_1 and q_1 with respect to interpretation I_{16} are $a + b$ and $c + d$, respectively. The interpretation for p_2 corresponds to:

$$I_{16}(p_2) = (I(p_1) * I(q_1)) \cdot m_2 = ((a + b) * (c + d)) \cdot m_2$$
$$= (a * c) \cdot m_2 \ + \ (a * d) \cdot m_2 \ + \ (b * c) \cdot m_2 \ + \ (b * d) \cdot m_2$$

This addition cannot be further simplified. Analogously, $I_{16}(q_2)$ can also be expressed as a sum of four sufficient causes – we just replace m_2 by n_2 in $I(p_2)$. But then, $I_{16}(p_3)$ corresponds to $(I_{16}(p_2) * I_{16}(q_2)) \cdot m_3$ and, applying distributivity, this yields a sum of 4×4 sufficient causes. In the general case, each atom p_n or q_n has $2^{2^{n-1}}$ sufficient causes so that expanding the complete causal value into this additive normal form becomes intractable. Furthermore, it also has been in (Cabalar et al. 2014b) that deciding whether a term without addition G is a brave necessary cause with respect to some regular program P is Σ_2^P-complete and, thus, deciding the existence of causal stable model is Σ_2^P-hard even for the class of programs that only contain a unique necessary causal literal.

Proposition 31 (From Cabalar et al. 2014b). *Given a causal term without addition $G \in \mathbf{C}_{Lb}$ and an causal term $t \in \mathbf{V}_{Lb}$ in which the right-hand operand of every application "$\cdot$" is a label, deciding whether $G \leq t$ is feasible in polynomial time.* $\square$

Proposition 32. *Let $\{t, u\} \subseteq \mathbf{V}_{Lb}$ be two causal term in which the right-hand operand of every application "$\cdot$" is a label. Then deciding whether $t \leq u$ is in* coNP. $\square$

Proof. Note hat $t \leq u$ iff every $G \in \mathbf{C}_{Lb}$ such that $G \leq t$ also satisfy $G \leq u$ which are decidable in polynomial time (Proposition 31). Consequently, deciding whether $t \leq u$ is coNP. □

Definition 32 (Causal graph). *Given a set of labels Lb, a* causal graph (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.* □

Theorem 10 (From (Fandinno 2015b)). *For any finite and definite program P with n rules,* $\mathrm{lfp}(T_P) = T_P^{\uparrow n}(\mathbf{0})$ *is its least model.* □

Definition 33. *Let P be a program and I be an interpretation. By* $\mathtt{simply-nec}(P^I)$ *we denote the program obtained from P^I by replacing every causal literal of the form $(\psi_{\mathcal{A}}^{nec} :: A)^{I(A)}$ by A if $I(A) \leq \sum \mathcal{A}$; and by 0 otherwise.* □

Lemma E.1. *Let P be a program and I be an interpretation. If $T_{P^I}^{\uparrow \alpha}(\mathbf{0}) \leq T_Q^{\uparrow \alpha}(\mathbf{0}) \leq I$, then $T_{P^I}^{\uparrow \alpha+1}(\mathbf{0}) \leq T_Q^{\uparrow \alpha+1}(\mathbf{0}) \leq I$ where $Q = \mathtt{simply-nec}(P^I)$.* □

Proof. Suppose first that $T_{P^I}^{\uparrow \alpha+1}(\mathbf{0})(A) \not\leq T_Q^{\uparrow \alpha+1}(\mathbf{0})(A)$ for some atom A. Then, since application and addition are $\leq$-monotonic, there must be some rule of the form of (A1) such that $T_{P^I}^{\uparrow \alpha}(\mathbf{0})(F) \not\leq T_Q^{\uparrow \alpha}(\mathbf{0})(F')$ where F' is just the result of replacing each causal literal of the form $(\psi_{\mathcal{A}}^{nec} :: A)^{I(A)}$ by A if $I(A) \leq \sum \mathcal{A}$; and by 0 otherwise. Since products and addition are monotonic, it is enough to show that

- $T_{P^I}^{\uparrow \alpha}(\mathbf{0})(\psi' :: A) \leq T_Q^{\uparrow \alpha}(\mathbf{0})(A)$ if $I(A) \leq \sum \mathcal{A}$, and
- $T_{P^I}^{\uparrow \alpha}(\mathbf{0})(\psi' :: A) = 0$ otherwise.

where

$$\psi'(G, I(A)) \ \stackrel{\text{def}}{=} \ \begin{cases} 1 & \text{iff exists some } G' \leq G \text{ s.t. } G' \in \max I(A) \text{ and } I(A) \leq \sum \mathcal{A} \\ 0 & \text{otherwise} \end{cases}$$

By definition,

$$T_{P^I}^{\uparrow \alpha}(\mathbf{0})(\psi_{\mathcal{A}}^{nec} :: A)^{I(A)} \ \stackrel{\text{def}}{=} \ \sum \big\{ G \in \max T_{P^I}^{\uparrow \alpha}(\mathbf{0})(A) \ \big| \ \psi'(G, I(A)) = 1 \big\}$$

One the one hand, $T_{P^I}^{\uparrow \alpha}(\mathbf{0})(\psi' :: A) \leq T_{P^I}^{\uparrow \alpha}(\mathbf{0})(A)$ holds for every causal literal $(\psi' :: A)$ and, by hypothesis, it holds that $T_{P^I}^{\uparrow \alpha}(\mathbf{0})(A) \leq T_Q^{\uparrow \alpha}(\mathbf{0})(A)$ and, therefore, $T_{P^I}^{\uparrow \alpha}(\mathbf{0})(\psi' :: A) \leq T_Q^{\uparrow \alpha}(\mathbf{0})(A)$ also holds. On the other hand, $I(A) \not\leq \sum \mathcal{A}$ implies that $\psi'(G, I(A)) = 0$ for every $G \in \mathbf{C}_{Lb}$ and, thus, $T_{P^I}^{\uparrow \alpha+1}(\mathbf{0})(\psi' :: A) = 0 \leq T_Q^{\uparrow \alpha+1}(\mathbf{0})(A)$.

Similarly, to show that $T_Q^{\uparrow \alpha}(\mathbf{0}) \leq I$ is enough to show $T_{P^I}^{\uparrow \alpha}(\mathbf{0})(A) \leq I(\psi' :: A)$ when $I(A) \leq \sum \mathcal{A}$. Note that, in case that $I(A) \not\leq \sum \mathcal{A}$, the causal literal $(\psi' :: A)$ has been replaced by 0. Then, for every $G \leq T_Q^{\uparrow \alpha}(\mathbf{0})(\psi' :: A)$ there is some $G' \in \max I(A)$ and $\psi'(G', I(A)) = 1$ and, consequently, it follows that $G \leq G' \leq I(\psi' :: A)$.

Furthermore, it is easy to see that $T_Q^{\uparrow \alpha}(\mathbf{0}) \leq I$ implies $T_Q^{\uparrow \alpha}(\mathbf{0})(A) \leq I(\psi' :: A)$ for every causal literal $(\psi' :: A)$. Just note that if $G \leq T_Q^{\uparrow \alpha}(\mathbf{0})(\psi' :: A)$, then $G \leq T_Q^{\uparrow \alpha}(\mathbf{0})(A) \leq I(A)$ and there is some $G' \leq G$ such that $G' \in \max I(A)$ and $I(A) \leq \sum \mathcal{A}$. Notice that facts $G \leq I(A)$, $G' \in \max I(A)$ and $G' \leq G$ implies that $G = G'$ and, thus, $G \in \max I(A)$. Therefore, $G \leq I(\psi' :: A)$ and, thus,

$$T_Q^{\uparrow \alpha}(\mathbf{0})(\psi' :: A) \ \leq \ I(\psi' :: A)$$

Note now that the evaluation of conjunctions and disjunctions is $\leq$-monotonic and, thus, it can be probed by induction that

$$T_{PI}^{\uparrow\alpha}(\mathbf{0})(F) \;\leq\; T_{Q}^{\uparrow\alpha}(\mathbf{0})(F) \;\leq\; I(F)$$

for every formula F. Finally, since addition and application are also $\leq$-monotonic, it can be shown by induction that

$$T_{PI}^{\uparrow\alpha}(\mathbf{0})(F){\cdot}r_i \;\leq\; T_{Q}^{\uparrow\alpha}(\mathbf{0})(F){\cdot}r_i \;\leq\; I(F){\cdot}r_i$$

and, thus,

$$T_{PI}^{\uparrow\alpha+1}(\mathbf{0})(A) \;\leq\; T_{Q}^{\uparrow\alpha+1}(\mathbf{0})(A) \;\leq\; I(A)$$

for every label $r_i \in Lb$ and atom $A \in At$. $\qquad\square$

Lemma E.2. *Let P be a program and I be an interpretation. Then $T_{PI}^{\uparrow\omega}(\mathbf{0}) \leq T_{Q}^{\uparrow\omega}(\mathbf{0}) \leq I$ where $Q = \mathtt{simply-nec}(P^I)$.* $\qquad\square$

Proof. By definition, $T_{PI}^{\uparrow 0}(\mathbf{0}) = T_{Q}^{\uparrow 0}(\mathbf{0}) = \mathbf{0} \leq I$ and, thus, by induction using Lemma E.1, it follows that $T_{PI}^{\uparrow\alpha}(\mathbf{0}) \leq T_{Q}^{\uparrow\alpha}(\mathbf{0}) \leq I$ for every successor ordinal α. For a limit ordinal α, $G \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})$ iff there is some $\beta < \alpha$ s.t. $G \leq T_{PI}^{\uparrow\beta}(\mathbf{0}) \leq T_{Q}^{\uparrow\beta}(\mathbf{0}) \leq T_{Q}^{\uparrow\alpha}(\mathbf{0})$ and, thus, $T_{PI}^{\uparrow\alpha}(\mathbf{0}) \leq T_{Q}^{\uparrow\alpha}(\mathbf{0})$. The proof of $T_{Q}^{\uparrow\alpha}(\mathbf{0}) \leq I$ is analogous. Hence, $T_{PI}^{\uparrow\omega}(\mathbf{0}) \leq T_{Q}^{\uparrow\omega}(\mathbf{0}) \leq I$. $\qquad\square$

Lemma E.3. *Let P be a program and I be an interpretation and $Q = \mathtt{simply-nec}(P^I)$. If for every atom A and causal term without addition $G \leq T_{Q}^{\uparrow\alpha}(\mathbf{0})(A)$ such that $G \in \max I(A)$, it holds that $G \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(A)$, then for every atom A and causal term without addition $G \leq T_{Q}^{\uparrow\alpha+1}(\mathbf{0})(A)$ such that $G \in \max I(A)$, it holds that $G \leq T_{PI}^{\uparrow\alpha+1}(\mathbf{0})(A)$* $\qquad\square$

Proof. Suppose there is some atom A and causal term without addition $G \in \mathbf{C}_{Lb}$ such that $G \leq T_{Q}^{\uparrow\alpha+1}(\mathbf{0})(A)$ and $G \in \max I(A)$, but $G \not\leq T_{Q}^{\uparrow\alpha+1}(\mathbf{0})(A)$. Then, since application and addition are $\leq$-monotonic, there must be some causal term without addition $G' \in \mathbf{C}_{Lb}$ and rule of the form of (A1) such that $G \leq G'{\cdot}r_i$ and $G' \leq T_{Q}^{\uparrow\alpha}(\mathbf{0})(F')$, but $G' \not\leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(F)$ where F' is just the result of replacing each causal literal of the form $(\psi_{\mathcal{A}}^{\mathrm{nec}} :: A)^{I(A)}$ by A if $I(A) \leq \sum \mathcal{A}$; and by 0 otherwise. Since products and addition are monotonic and every causal literal in F of the form of $(\psi' :: A)$ is replaced by 0 in F' when $I(A) \not\leq \sum \mathcal{A}$, it is enough to show that

- $G \leq T_{Q}^{\uparrow\alpha}(\mathbf{0})(A)$ and $G \in \in \max I(A)$ implies $G \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(\psi' :: A)$ when $I(A) \leq \sum \mathcal{A}$

where $\psi' \overset{\text{def}}{=} (\psi_{\mathcal{A}}^{\mathrm{nec}})^{I(A)}$. Indeed, by hypothesis, from $G \leq T_{Q}^{\uparrow\alpha}(\mathbf{0})(A)$ and $G \in \max I(A)$ it follows that $G \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(A)$. Furthermore, $G \in \max I(A)$ and $I(A) \leq \sum \mathcal{A}$ also imply that $(\psi'(G, I(A)) = 1$ holds and, consequently, it follows that $G \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})(\psi' :: A)$. $\qquad\square$

Lemma E.4. *Let P be a program and I be an interpretation and $Q = \mathtt{simply-nec}(P^I)$. Then, $I = T_{Q}^{\uparrow\omega}(\mathbf{0})$ iff $I = T_{PI}^{\uparrow\omega}(\mathbf{0})$* $\qquad\square$

Proof. First, assume that $I = T_{PI}^{\uparrow\omega}(\mathbf{0})$. From Lemma E.2, it follows that $T_{PI}^{\uparrow\omega}(\mathbf{0}) \leq T_{Q}^{\uparrow\omega}(\mathbf{0}) \leq I$ and, thus, $I = T_{PI}^{\uparrow\omega}(\mathbf{0})$ implies $I = T_{Q}^{\uparrow\omega}(\mathbf{0})$.

The other way around. Assume $I = T_{Q}^{\uparrow\omega}(\mathbf{0})$ and assume as induction hypothesis that $G \leq T_{Q}^{\uparrow\beta}(\mathbf{0})(A)$ and $G \in \max I(A)$, imply $G \leq T_{PI}^{\uparrow\beta}(\mathbf{0})(A)$ for every ordinal $\beta < \alpha$. By definition, $T_{Q}^{\uparrow 0}(\mathbf{0}) = \mathbf{0}$ and the hypothesis holds vacuous. Furthermore, using Lemma E.3, the hypothesis holds for every successor ordinal α. For a limit ordinal α, $G \leq T_{Q}^{\uparrow\alpha}(\mathbf{0})$ iff there is some $\beta < \alpha$

s.t. $G \leq T_Q^{\uparrow\beta}(\mathbf{0})$ and, thus, $G \leq T_{PI}^{\uparrow\beta}(\mathbf{0}) \leq T_{PI}^{\uparrow\alpha}(\mathbf{0})$. Then, for every $G \leq I(A) = T_Q^{\uparrow\omega}(\mathbf{0})$ there is some $G' \in \mathbf{C}_{Lb}$ such that $G' \in \max I(A) = T_Q^{\uparrow\omega}(\mathbf{0})$ and, thus, $G \leq G' \leq T_{PI}^{\uparrow\omega}(\mathbf{0})(A)$. That is, $T_Q^{\uparrow\omega}(\mathbf{0}) \leq T_{PI}^{\uparrow\omega}(\mathbf{0})$. Finally, from Lemma E.2, it follows that $T_{PI}^{\uparrow\omega}(\mathbf{0}) \leq T_Q^{\uparrow\omega}(\mathbf{0})$ and, thus, it also holds $I = T_{PI}^{\uparrow\omega}(\mathbf{0})$. $\qquad\square$

Lemma E.5. *Let P be a program and I be an interpretation and $Q = \mathtt{simply-nec}(P^I)$. Then, I is a causal stable model of P iff $T_Q^{\uparrow\omega}(\mathbf{0}) = I$.* $\qquad\square$

Proof. By definition, I is a causal stable model of P iff I is the least model of P^I iff $T_{PI}^{\uparrow\omega}(\mathbf{0}) = I$ (Theorem 3) iff $T_{PI}^{\uparrow\omega}(\mathbf{0}) = I$ (Lemma E.4). $\qquad\square$

Proposition 33. *Let t be a term and $\mathcal{A}$ be a set of labels. Then, $t \leq \sum \mathcal{A}$ is decidable in polynomial time.* $\qquad\square$

Proof. If $t \in Lb$ is a label, then $t \leq \sum \mathcal{A}$ iff $t \in \mathcal{A}$ which is clearly decidable in polynomial time. Otherwise, we assume as induction hypothesis that $u \leq \sum \mathcal{A}$ and $w \leq \sum \mathcal{A}$ are decidable in polynomial time for every subterms u and w of t. In case that $t = u + w$, then $t \leq \sum \mathcal{A}$ iff $u \leq \sum \mathcal{A}$ and $w \leq \sum \mathcal{A}$ which are both decidable in polynomial time. Similarly, in case that $t = u * w$ or $t = u \cdot w$, then $t \leq \sum \mathcal{A}$ iff $u \leq \sum \mathcal{A}$ or $w \leq \sum \mathcal{A}$ which are both decidable in polynomial time. $\qquad\square$

Proposition 34. *Let P be a causal program containing only necessary causal literals. Then, deciding whether there exists a causal stable model of P or not is in* NP. $\qquad\square$

Proof. First, note that there exists some causal stable model I of P iff there must exists some program Q and casual stable model I such that Q is the result of replacing every maximal subformula in P of the form *not E* by 1 if $I \models not\ E$ and by 0 otherwise. Just note that $Q^I = P^I$ for every interpretation I. Then, from Lemma E.5, I is a causal stable model of P iff I is a causal stable model of Q iff $T_{Q'}^{\uparrow\omega}(\mathbf{0}) = I$ where $Q' = \mathtt{simply-nec}(P^I)$.

Hence, instead of guessing an interpretation I we will guess a program Q'. Let Q for every maximal subformula be the result of replacing every maximal subformula in P of the form *not E* by a guessed 0 or 1 and let Q' be he result of replacing every necessary causal literal in Q of the form of $(\psi_{\mathcal{A}}^{\mathrm{nec}} :: A)$ by a guessed 0 or A. Note that, since P only contains necessary causal literals, Q' is a positive regular (hence monotonic) program. From Theorem 10, it follows that $T_{Q'}^{\uparrow n}(\mathbf{0}) = T_{Q'}^{\uparrow\omega}(\mathbf{0})$ is the least fixpoint of $T_{Q'}$ and the least model of Q' where n is the number of rules in Q, which is the same as the number of rules in P. Let us define $I = T_{Q'}^{\uparrow n}(\mathbf{0})$. Since Q' is a regular program each step of $T_{Q'}$ only involves the creation of a term from its subterms, which is feasible in polynomial time and, thus, I can be computable in polynomial time.

Let us now check whether $Q' = \mathtt{simply-nec}(P^I)$. Then, $\mathtt{fail}$ if $I = T_{Q'}^{\uparrow n}(\mathbf{0})$ do not satisfy one of the following conditions

- $I \models not\ E$ for some maximal subformula whose guessed value was 0
- $I \not\models not\ E$ for some maximal subformula whose guessed value was 1
- $I(A) \leq \sum \mathcal{A}$ for some necessary causal literal $(\psi_{\mathcal{A}}^{\mathrm{nec}} :: A)$ whose guessed value was 0
- $I(A) \not\leq \sum \mathcal{A}$ for some necessary causal literal $(\psi_{\mathcal{A}}^{\mathrm{nec}} :: A)$ whose guessed value was A

If reached this point, then $Q' = \mathtt{simply-nec}(P^I)$ and, hence, we the procedure $\mathtt{succeed}$. It just remain to show that these four conditions can be checked in polynomial time. The two first only involve checking whether $I(E) = 0$ which is feasible simply simplifying the obtained causal term and looking whether it is 0 or not. Finally, since $\mathcal{A} \subseteq Lb$ is a set of labels, from Proposition 33, it follows that $I(A) \leq \sum \mathcal{A}$ can be checked in polynomial time. $\qquad\square$

Proposition 35. *Let P be a causal program containing only necessary causal literals. Then, deciding whether there exists a causal stable model of P or not is in* NP-*complete (it is* NP-*hard even in P only contains a single negated regular literal or P is positive but contains a single constraint).* □

Proof. NP membership follows directly from Proposition 34 while NP-hard follows from the fact that every regular program in also a causal program and deciding the existence of stable model for standard programs in NP-complete. To show that it is NP-hard even when P only contains a negated regular literal, we reduce the existence of stable model for standard program to the satisfiability of a CNF Boolean formula φ to the existence of causal stable model of a program P. We assume without of generality that no clause in φ has complementary variables. For every variable x_k occurring in φ, let P_k be a program containing rules of the form

$$
\begin{aligned}
x_k &: & x_k &\leftarrow \\
t_{x_k} &: & x_k &\leftarrow \mathcal{A}_{tk} \ \texttt{necessary for } x_k \\
f_{x_k} &: & x_k &\leftarrow \mathcal{A}_{fk} \ \texttt{necessary for } x_k
\end{aligned}
$$

where $\mathcal{A}_{tk} = \{t_{x_k}, x_k\}$ and $\mathcal{A}_{fk} = \{f_{x_k}, x_k\}$. For each clause c_j in φ, let P'_j be a program containing a rule of the form of

$$
c_j \leftarrow \mathcal{A}_{jk} \ \texttt{necessary for } x_k \tag{E1}
$$

for each variable x_k in c_j, where $\mathcal{A}_{jk} = \{t_{x_k}, x_k\}$ if x_k occurs positively in the clause c_j and $\mathcal{A}_{jk} = \{f_{x_k}, x_k\}$ if x_k occurs negatively in c_j and let P'' be a program containing the following rule

$$
p \ \leftarrow \ c_1, \ldots, c_m
$$

where $c_1, \ldots, c_m$ are all the clauses in φ. Note that no atom occurring in the body of a program P_k occurs in the head of a program P'_j nor P'' and no atom occurring in the body of a program P'_j occurs in the head of a program P''. Hence, we can use the Splitting Theorem (Theorem 5).

   Each program P_k has two causal stable models I_k and J_k that satisfy $I_k(x_k) = x_k + t_{x_k}$ and $J_k(x_k) = x_k + f_{x_k}$ and, thus, $P = \bigcup_{k \leq n} P_n$ has 2^n causal stable models here n is the number of variables in φ: each causal stable model I satisfying $I(x_k) = x_k + t_{x_k}$ or $I(x_k) = x_k + f_{x_k}$ for each variable x_k. We say that an variable x_k is true in an interpretation I if $I(x_k) = x_k + t_{x_k}$ and that is false if $I(x_k) = x_k + f_{x_k}$. Then, $(P \cup P'_j)$ also has 2^n causal models models where each causal stable model I satisfy

$$
\begin{aligned}
I(c_j) \ = \ & \sum \{\, x_k + t_{x_k} \mid I(x_k) = x_k + t_{x_k} \text{ and } x_k \text{ occurs positively in } c_j \,\} \\
+ \ & \sum \{\, x_k + f_{x_k} \mid I(x_k) = x_k + t_{x_k} \text{ and } x_k \text{ occurs negatively in } c_j \,\}
\end{aligned}
$$

That is, $I(c_j) \neq 0$ iff there is some variable x_k such that x_k is true in I and occurs positively in c_j or there is some variable x_k such that x_k is false in I and occurs negatively in c_j iff I represents an assignment that satisfies the clause c_j. Let $P' = \bigcup_{j \leq m} P'_j$. Then, P' also has 2^n causal models models: each causal stable model I satisfy for each clause c_j that $I(c_j) \neq 0$ iff represents an assignment that satisfies the clause c_j. It is easy to see now that $(P' \cup P'')$ has 2^n causal stable models where $I(p) \neq 0$ iff every c_j satisfy $I(c_j) \neq 0$ iff I represents an assignment that satisfy all clauses c_j in φ iff I represents an assignment that satisfy φ. Finally, let P be the

result of adding, to the program $(P' \cup P'')$, the following rule

$$p \leftarrow not\, p$$

Then, P has a causal stable model iff there is a causal stable model I of $(P' \cup P'')$ such that $I(p) \neq 0$ iff I represents an assignment that satisfy φ. Alternatively, let Q' be the result of adding, to the program $(P' \cup P'')$, the following rules

$$q: \quad q \leftarrow$$
$$q \leftarrow p$$

and Q be the result of adding to Q' the constraint

$$\bot \leftarrow \mathcal{A}_p \; \texttt{necessary for} \; q$$

where $\mathcal{A}_p = \{q\}$. Then, Q' has 2^n causal stable models: each causal stable model I satisfying $I(q) = q + I(p)$ and Q has a causal stable model I iff there is a causal stable model I of Q' such that $I(q) > q$ iff there is a causal stable model I of Q' such that $I(p) \neq 0$ iff I represents an assignment that satisfy φ. $\square$