Causal Graph Justifications of Logic Programs∗

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

Michael Fink
Vienna University of Technology,
Institute for Information Systems
Vienna, Austria
(*e-mail:* fink@kr.tuwien.ac.at)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

In this work we propose a multi-valued extension of logic programs under the stable models semantics where each true atom in a model is associated with a set of justifications. These justifications are expressed in terms of *causal graphs* formed by rule labels and edges that represent their application ordering. For positive programs, we show that the causal justifications obtained for a given atom have a direct correspondence to (relevant) syntactic proofs of that atom using the program rules involved in the graphs. The most interesting contribution is that this causal information is obtained in a purely semantic way, by algebraic operations (product, sum and application) on a lattice of causal values whose ordering relation expresses when a justification is stronger than another. Finally, for programs with negation, we define the concept of *causal stable model* by introducing an analogous transformation to Gelfond and Lifschitz's program reduct. As a result, default negation behaves as "absence of proof" and no justification is derived from negative literals, something that turns out convenient for elaboration tolerance, as we explain with several examples.

KEYWORDS: Answer Set Programming, Causality, Knowledge Representation, Multi-valued 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 reading for default literal *not p* is "there is no way to derive *p*." Although this idea seems quite intuitive, it actually resorts to a concept, the *ways to derive p*, outside the scope of existing LP semantics. In other words, LP semantics point out whether there exists some derivation for an atom, but do not provide the derivations themselves, if several alternatives exist.

However, such information on justifications for atoms can be of great interest for Knowledge Representation (KR), and especially, for dealing with problems related to causality. An

∗ This research was partially supported by Spanish MEC project TIN2009-14562-C05-04, Xunta program INCITE 2011 and Inditex-University of Corunna 2013 grants, as well as by the Austrian Science Fund (FWF) project P24090.

important challenge in causal reasoning is the capability of not only deriving facts of the form "A has caused B," but being able to represent them and reason about them. This kind of information is crucial, for instance, in diagnosis or in legal reasoning. As an example, take the assertion:

"If somebody causes an accident, (s)he is legally responsible for that."

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. Regarding representation of the above law, for instance, one might think of an informal rule:

$$responsible(X,Y) \leftarrow action(A), person(X), accident(Y), \text{“}do(A,X) \texttt{ caused } occurs(Y)\text{”}.$$

If the pseudo-literal "$do(A,X)$ `caused` $occurs(Y)$" actually corresponds to an explicit representation of all the possible ways of causing an accident, however, one immediately runs into a problem of *elaboration tolerance* (McCarthy 1998) — adding new rules that causally connect $do(A,X)$ to $occurs(Y)$ (in a direct or indirect way) would force us to build new rules for $responsible(X,Y)$. What is needed instead, and what we actually propose, is to introduce, indeed, some kind of new LP literal "A caused B," with *an associated semantics* capable of revealing causes A of a given true atom B.

While not straightforward, the rewarding perspective of such a semantic approach is an extension of Answer Set Programming (ASP) (Brewka et al. 2011) with causal literals capable of representing different kinds of causal influences (sufficient cause, necessary cause, etc). In this paper, we tackle the above issue and, as a first step and basic underlying requirement, develop a suitable semantics capable of associating causal justifications with each true atom. To this end, we propose a multi-valued extension of logic programs under the stable model semantics (Gelfond and Lifschitz 1988) where each true atom in a model is associated with a set of justifications in the form of *causal graphs*. To further illustrate our motivation, consider the following example.

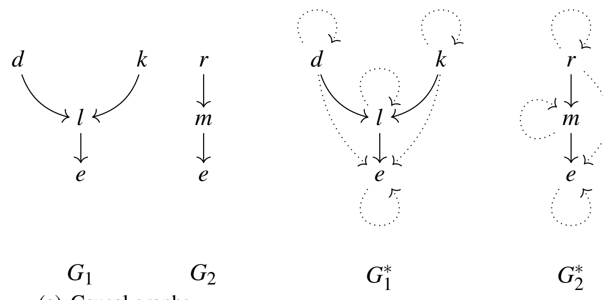
Example 1 (*From Cabalar 2011*)
Some country has a law l that asserts that driving drunk is punishable with imprisonment. On the other hand, a second law m specifies that resisting arrest has the same effect. The execution e of a sentence establishes that a punishment implies imprisonment. Suppose that some person drove drunk and resisted to be arrested. □

We can capture this scenario with the following logic program P_1:

$$\begin{array}{rlrlrl} l: & punish \leftarrow drive, drunk & m: & punish \leftarrow resist & e: & prison \leftarrow punish \\ d: & drive & k: & drunk & r: & resist \end{array}$$

The least model of this positive program makes atom *prison* true, so we know that there exists a possible derivation for it. In particular, two alternative justifications can be made, corresponding to the graphs in Figure 1(a): driving drunk and, independently, resisting to authority. Rather than the result of syntactic transformations, in our approach these graph structures are embodied in the semantics of the logic programs. More specifically, we summarise our contributions as follows.

- We define a multi-valued semantics for (normal) logic programs based on causal graphs. An important result is that, despite of this semantic nature, we are able to show that causal values have a direct correspondence to (relevant) syntactic proofs using the program rules involved in the graphs (cf. Section 4).



(a) Causal graphs (b) Reflexive, transitive closures

Fig. 1. Derivations G_1 and G_2 justifying atom *prison* in program P_1.

- We also define an ordering relation that specifies when a cause is *stronger* than another, and show how causal values form a lattice with three associated algebraic operations: a product '$*$' representing conjunction or joint causation; a sum '$+$' representing alternative causes; and a non-commutative product '$\cdot$' that stands for rule application. We study beneficial properties of these operations that allow manipulating and reasoning with causal values in an analytical way (cf. Sections 2 and 3).
- Finally, we provide several examples illustrating the behaviour of the semantics on typical KR scenarios and action theories (throughout the paper; cf. also Appendix B).

Fostered by its algebraic treatment of causal values, our work facilitates the incorporation of dedicated, more specific causal expressions representing causal influence of different kinds.

2 Causes as graphs

In this and subsequent Section 3, we introduce the lattice of causal values in two different steps. In a first step, we focus on the idea of an "individual" cause and then we proceed to explain the concept of causal value that allows collecting different alternative causes.

We begin recalling several graph definitions and notation. A (directed) *graph* is a pair $\langle V,E\rangle$ where V is a set of vertices $V \subseteq Lb$ and E is a set of edges $E \subseteq V \times V$. In the following definitions, let $G = \langle V,E\rangle$ and $G' = \langle V',E'\rangle$ be two graphs. We say that G is a *subgraph* of G', written $G \subseteq G'$, when $V \subseteq V'$ and $E \subseteq E'$. We write $G \cup G'$ to stand for the graph $\langle V \cup V', E \cup E'\rangle$. We represent the reflexive and transitive closure of G as G^*. Finally, we introduce a *concatenation* operation $G \odot G'$ on graphs corresponding to a graph with vertices $V \cup V'$ and edges $E \cup E' \cup \{(x,y) \mid x \in V, y \in V'\}$. Notice that, $G \cup G' \subseteq G \odot G'$, that is, the concatenation extends the union of graphs by adding all possible arcs that go from some node in G to some node in G'.

Let Lb be some finite set of (rule) labels. A *causal graph* $G = \langle V,E\rangle$ is just a graph whose vertices are labels, that is, $V \subseteq Lb$. Intuitively, the vertices correspond to rules involved in a derivation of a given atom (or formula), and the edges point out a (partial) ordering of application of rules in the derivation. Figure 1(a) shows two causal graphs with labels from P_1.

Definition 1 (*Cause*)
A *cause* G is a causal graph closed under reflexivity and transitivity, i.e., $G^* = G$. For a set of labels Lb, we denote with $\mathbf{C}_{Lb}$ the set of all possible causes over Lb. □

The intuition is that a cause captures not only the direct application of rules in a justification, but also a dependence relation among them. E.g., to form a cause, G_2 in Figure 1(a) would also include an arc (r,e) meaning that the application of rule e for *prison* depends on r for *resist*. Reflexivity is convenient for simpler definitions. For instance, the cause formed by a single label l also has a single edge (l,l)—we call this an *atomic* cause and represent it just by its label. For simplicity, we will usually omit transitive and reflexive arcs when depicting a cause, that is we use causal graphs for representing their reflexive and transitive closure. For instance, taking G_1 and G_2 in Figure 1(a) as causes actually amounts to considering the graphs shown in Figure 1(b), where previously omitted arcs are shown as dotted lines. We define next a natural ordering relation among causes.

Definition 2 (*Sufficient cause*)
A cause G is *sufficient* for another cause G', written $G \leq G'$, when $G \supseteq G'$. □

Saying that G is sufficient for G' intuitively means that G contains enough information to yield the same effect than G', but perhaps more than needed (this explains $G \supseteq G'$). For this reason, we sometimes read $G \leq G'$ as "G' is *stronger* than G." For instance, cause G_2 for *prison* is obviously sufficient for justifying atom *punish*, although e is not necessary for that purpose and can be removed, so that arc (r,m) is actually enough.

Since graphs with the subgraph relation form a poset, the set of causes also constitutes a poset $\langle \mathbf{C}_{Lb}, \leq \rangle$ with a top element corresponding to the empty cause, that is, the empty graph $G_\emptyset = \langle \emptyset, \emptyset \rangle$. This cause stands for a kind of "absolute truth" and is of interest for including rules or facts one does not want to label, that is, their effect will not be traced in the justifications.

Any cause can be built up from labels (atomic causes) using two basic operations: the product $G * G' \overset{\text{def}}{=} (G \cup G')^*$ that stands for union of causes or *joint interaction*, and the concatenation $G \cdot G' \overset{\text{def}}{=} (G \odot G')^*$ that captures their *sequential application*. The reason for applying a closure is that the result of $G \cup G'$ and $G \odot G'$ on causes does not need to be closed under transitivity. We can extend the product to any (possibly infinite) set of causes S so that $\Pi\, S \overset{\text{def}}{=} \left(\bigcup_{G \in S} G \right)^*$.

Example 2 (*Ex. 1 continued*)
The cause for the body of l in P_1 is the product of causes for *drive* and *drunk*, that is $d * k$ formed with vertices $\{d,k\}$ and edges $\{(d,d),(k,k)\}$. As a result, the explanation of the rule head, *punish*, is formed by the concatenation of its body cause $d * k$ with its label, that is $(d*k) \cdot l$. In its turn, this becomes the cause for the body of e and so, we get the explanation $(d*k) \cdot l \cdot e$ for atom *prison* represented as G_1 in Figure 1(b). Similarly, G_2 corresponds to $r \cdot m \cdot e$. □

When writing these causal expressions, we assume that '$\cdot$' has higher priority than '$*$'. Furthermore, we will usually omit '$\cdot$' when applied to consecutive labels, so that $r \cdot m \cdot e$ will be abbreviated as *rme*. It is easy to see that $G * G' = G' * G$ while, in general, $G \cdot G' \neq G' \cdot G$, that is, concatenation is not commutative. Another observation is that $G \cdot G' \leq G * G'$, that is, concatenation is sufficient for the product, but the opposite does not hold in general. Moreover, in our running example, we can check that $(d*k) \cdot l$ is equal to $(d \cdot l) * (k \cdot l)$. In fact, application distributes over products and, as a result, we can identify any cause with the product of all its edges. To conclude this section, we note that the set of causes $\mathbf{C}_{Lb}$ ordered by $\leq$ forms a lower semilattice $\langle \mathbf{C}_{Lb}, * \rangle$, where the product constitutes the infimum.

3 Alternative causes

Having settled the case for individual causes, let us now proceed to represent situations in which several alternative and independent causes can be found for an atom p. The obvious possibility is just using a *set of causes* for that purpose. However, we should additionally disregard causes for which a stronger alternative exists. For instance, as we saw before, cause *rme* is sufficient for *punish* and therefore, it is also an alternative way to prove this atom, but redundant in the presence of the stronger cause *rm*. This suggests to choose sets of $\leq$-maximal causes as our appropriate 'truth values'. In principle, this is the central idea, although $\leq$-maximal causes incur some minor inconveniences in mathematical treatment. For instance, the union of two sets of maximal causes, does not need to be a set of maximal causes. Besides, the operations of product and concatenation are expected to extend to the sets adopted as causal values. To address these issues, a more solid representation is obtained resorting to *ideals* of causes, as we see next.

Given any poset $\langle A, \leq \rangle$, an *ideal* I is any set $I \subseteq A$ satisfying[1]: if $x \in I$ and $y \leq x$ then $y \in I$. A compact way of representing an ideal I is by using its set of maximal elements S, since the rest of I contains *all elements below* them. The *principal ideal* of an individual element $x \in A$ is denoted as $\downarrow x \stackrel{\text{def}}{=} \{y \in A \mid y \leq x\}$. We extend this notion for any set of elements S so that $\downarrow S \stackrel{\text{def}}{=} \bigcup \{\downarrow x \mid x \in S\} = \{y \in A \mid y \leq x, \text{for some } x \in S\}$. Thus, we will usually represent an ideal I as $\downarrow S$ where S are the maximal elements in I. In fact, maximal elements constitute the relevant information provided by the ideal, while keeping all other elements is convenient for simplicity of algebraic treament (but we do not assign a particular meaning to them).

Definition 3 (*Causal Value*)
Given a set of labels *Lb*, a *causal value* is any ideal for the lower semilattice $\langle \mathbf{C}_{Lb}, * \rangle$. We denote by $\mathbf{V}_{Lb}$ the set of causal values. □

The product of causes maps to the intersection of causal values, that is $\downarrow (G_1 * G_2) = \downarrow G_1 \cap \downarrow G_2$. Analogously, the ordering relation $\leq$ among causes maps to the subset relation $\subseteq$ among causal values, i.e. $G_1 \leq G_2$ iff $\downarrow G_1 \subseteq \downarrow G_2$. In fact, for ideals U, U' we will keep the notation $U \leq U'$ to stand for $U \subseteq U'$. Concatenation also extends easily to any pair U, U' of causal values as: $U \cdot U' \stackrel{\text{def}}{=} \downarrow \{G \cdot G' \mid G \in U \text{ and } G' \in U'\}$. Finally, the union of causal values allows for collecting alternative causes stemming from different rules.

Example 3 (*Ex. 1 continued*)
The interpretation for *punish* has two alternative causes $(d*k) \cdot l$ and *rm* that become the causal values $\downarrow (d*k) \cdot l$ and $\downarrow rm$. The causal value for *punish* is then formed by their union:

$$\downarrow (d*k) \cdot l \ \cup \ \downarrow rm \ = \downarrow \{\ (d*k) \cdot l, \ rm \ \}$$

This ideal contains, among others, the cause *rme*, although it is not maximal due to *rm*:

$$\downarrow \{\ (d*k) \cdot l, \ rm \ \} \ \cup \downarrow rme \ = \downarrow \{\ (d*k) \cdot l, \ rm, \ rme\} \ = \downarrow \{\ (d*k) \cdot l, \ rm \ \} \qquad \square$$

Theorem 1
$\langle \mathbf{V}_{Lb}, \cup, \cap \rangle$ is the free completely distributive lattice generated by the lower semilattice $\langle \mathbf{C}_{Lb}, * \rangle$ using the injective homomorphism (or embedding) $\downarrow : \mathbf{C}_{Lb} \longrightarrow \mathbf{V}_{Lb}$. □

The lattice of causal values has as bottom element the empty ideal $\emptyset$ (standing for "falsity") and as top element the ideal formed by the empty cause $\downarrow G_{\emptyset}$ which corresponds to the whole set of causes $\mathbf{C}_{Lb}$ (standing for "absolute truth"). By abuse of terminology we will stick to the term "cause" referring to any causal value of the form $\downarrow G$ where G is a cause. To improve

[1] We use terminology from (Stumme 1997). In some texts this is known as *semi-ideal* or *down-set* to differentiate this definition from the stronger case in which ideals are applied on a (full) lattice rather than a semi-lattice.

readability, we introduce the syntactic notion of *causal term*, that allows for representing causal values without explicitly resorting to graphs or ideals.

Definition 4 (*Causal term*)
A *causal term*, t, over a set of labels Lb, is recursively defined as one of the following expressions:

$$t ::= l \mid \prod S \mid \sum S \mid t_1 \cdot t_2$$

where $l \in Lb$, t_1, t_2 are in their turn causal terms and S is a (possibly empty) set of causal terms. When S is finite and non-empty, $S = \{t_1, \ldots, t_n\}$ we write $\prod S$ simply as $t_1 * \cdots * t_n$ and $\sum S$ as $t_1 + \cdots + t_n$. By 1 and 0 we respectively denote $\prod \emptyset$ and $\sum \emptyset$. □

As usual, we assume that '$*$' has higher priority than '$+$'. Each causal term t represents a causal value, $value(t)$ naturally defined as follows:

$$\begin{array}{rcl} value(l) & \overset{\text{def}}{=} & \downarrow l \\ value\left(\sum S\right) & \overset{\text{def}}{=} & \bigcup \{\, value(t) \mid t \in S \,\} \end{array} \qquad \begin{array}{rcl} value(t_1 \cdot t_2) & \overset{\text{def}}{=} & value(t_1) \cdot value(t_2) \\ value\left(\prod S\right) & \overset{\text{def}}{=} & \bigcap \{\, value(t) \mid t \in S \,\} \end{array}$$

The value for label l is the ideal for the cause l which, as we commented before, is an abbreviation for the graph formed by vertex l and edge (l,l). When $S = \emptyset$, the union of elements in S corresponds to $\emptyset$ (the identity for the $\cup$ operation) that is, $value(0) = value(\sum \emptyset) = \bigcup \emptyset = \emptyset$. Analogously, the intersection of elements in $S = \emptyset$ corresponds to $\mathbf{C}_{Lb}$ (the set of all possible elements, i.e., the identity for $\cap$ in this context), i.e., $value(1) = value(\prod \emptyset) = \bigcap \emptyset = \mathbf{C}_{Lb} = \downarrow G_\emptyset$.

From now on, we will use causal terms as compact representations of causal values. Causes correspond to causal terms without addition (note that this also excludes 0, the empty sum). Several interesting algebraic properties can be proved for causal values. In particular, Theorem 1 guarantees that they form a completely distributive lattice with respect to '$*$' and '$+$' satisfying the standard properties such as associativity, commutativity, idempotence, absorption or distributivity on both directions. Besides, as usual, 0 (resp. 1) is the annihilator for '$*$' (resp. '$+$') and the identity for '$+$' (resp. '$*$'). More significantly, the main properties for '$\cdot$' are shown in Figure 2.

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

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

Fig. 2. Properties of the '$\cdot$' operator (c, d, e represent causes).

4 Positive programs and minimal models

Let us now reconsider logic programs and provide a semantics based on the causal values we have just defined. For the syntax, we recall standard LP definitions, just slightly extending it by introducing rule labels. A *signature* is a pair $\langle At, Lb \rangle$ of sets that respectively represent a set of *atoms* (or *propositions*) and a set of *labels*. As usual, a *literal* is defined as an atom p (positive literal) or its default negation *not* p (negative literal). In this paper, we will concentrate on programs without disjunction in the head (leaving its treatment for future work).

Definition 5 (*Causal logic program*)
Given a signature $\langle At, Lb \rangle$, a *(causal) logic program* P is a set of rules of the form:

$$t : H \leftarrow B_1, \dots, B_n, \tag{1}$$

where t is a label l over Lb or the constant 1, H is an atom or $\perp$ (the *head* of the rule) and $B_1, \dots, B_n$ are literals (the *body* of the rule). □

For any rule R of the form (1) we define $label(R) \overset{\text{def}}{=} t$. We denote by $head(R) \overset{\text{def}}{=} H$ its *head*, and by $body(R) \overset{\text{def}}{=} \{B_1, \dots, B_n\}$ its *body*. When $n = 0$ we say that the rule is a *fact* and omit the symbol '$\leftarrow$.' When $t \in Lb$ we say that the rule is labelled; otherwise $t = 1$ and we omit both t and ':'. By these conventions, for instance, an unlabelled fact p is actually an abbreviation of $(1 : p \leftarrow)$. A logic program P is *positive* if it contains no default negation. A program is *uniquely labelled* if no pair of labelled rules share the same label, and *completely labelled* if, additionally, all rules are labelled. For instance, P_1 is completely labelled.

Given a signature $\langle At, Lb \rangle$ a *causal interpretation* is a mapping $I : At \ \rightarrow \ \mathbf{V}_{Lb}$ assigning a causal value to each atom. An interpretation is *classical* if it maps all atoms to $\{0, 1\}$. Furthermore, for any causal interpretation we can define its corresponding classical interpretation, written I^{cl}, so that, for any atom p: $I^{cl}(p) \overset{\text{def}}{=} 0$ if $I(p) = 0$; and $I^{cl}(p) \overset{\text{def}}{=} 1$ otherwise. The partial order $\leq$ is extended to interpretations I, J by $I \leq J \overset{\text{def}}{=} I(p) \leq J(p)$ for each atom $p \in At$. Hence, there is a $\leq$-bottom interpretation $\mathbf{0}$ (resp. a $\leq$-top interpretation $\mathbf{1}$) that stands for the interpretation mapping each atom p to 0 (resp. 1). The value assigned to a negative literal *not* p by an interpretation I, denoted as $I(not\ p)$, is defined as: $I(not\ p) \overset{\text{def}}{=} 1$ if $I(p) = 0$; and $I(not\ p) \overset{\text{def}}{=} 0$ otherwise.

Definition 6 (*Causal model*)
Given a positive causal logic program P, a causal interpretation I is a *causal model*, in symbols $I \models P$, if and only if, for each rule $R \in P$ of the form (1), the following condition holds:

$$\big(\ I(B_1) * \dots * I(B_n) \ \big) \cdot t \leq I(H)$$ □

Example 4 (*Ex. 1 continued*)
Take rule l from program P_1 and let I be such that $I(drive) = d$ and $I(drunk) = k$. Then I will be a model of l when $(d * k) \cdot l \leq I(punish)$. In particular, this holds when $I(punish) = (d * k) \cdot l + r \cdot m$ which was the value we expected for that atom. But it would also hold when, for instance, $I(punish) = l + m$ or $I(punish) = 1$. The inequality in Definition 6 is important to accommodate possible additional facts such as $(l : punish)$ or even $(1 : punish)$ in the program. □

The fact that any $I(punish)$ greater than $(d * k) \cdot l + r \cdot m$ also becomes a model clearly points out the need for selecting *minimal* models. In fact, as it is the case for non-causal programs, positive programs have a $\leq$-least model that can be computed by iterating an extension of the well-known *direct consequences operator* (van Emden and Kowalski 1976).

Definition 7 (*Direct consequences*)
Given a positive logic program P over signature $\langle At, Lb \rangle$, the operator of *direct consequences* is a function $T_P : \mathbf{I} \longrightarrow \mathbf{I}$ such that, for any causal interpretation I and any atom $p \in At$:

$$T_P(I)(p) \ \overset{\text{def}}{=} \ \sum \big\{ \ \big(\ I(B_1) * \dots * I(B_n) \ \big) \cdot t \ \mid \ (t : p \leftarrow B_1, \dots, B_n) \in P \ \big\}$$

Theorem 2
Let P be a positive logic program P over signature $\langle At, Lb\rangle$. Then, (*i*) there exists some integer $n \geq 0$ such that $T_P \uparrow^{\omega} (\mathbf{0}) = T_P \uparrow^{n} (\mathbf{0}) = \mathit{lfp}(T_P)$, and (*ii*) $\mathit{lfp}(T_P)$ is the least model of P. □

The proof of this theorem relies on an encoding of causal logic programs into *Generalized Annotated Logic Programming* (GAP) (Kifer and Subrahmanian 1992). GAP is a general multi-valued framework for LP for which the main properties for the least model and T_P operator have been already proved. Theorem 2 is an interesting result, but it does not explain the relation between the causal values we obtain for atoms and their role in the program. We illustrate next that there exists a direct relation between causal values and the idea of *proof* from a positive program.

Definition 8
Given a positive program P, a *proof* $\pi(p)$ of an atom p can be recursively defined as a derivation:

$$\pi(p) \quad \overset{\text{def}}{=} \quad \frac{\pi(B_1)\ \ldots\ \pi(B_n)}{p}\ (R),$$

where $R \in P$ is a rule with $\mathit{head}(R) = p$ and $\mathit{body}(R) = \{B_1, \ldots, B_n\}$. When $n = 0$, the derivation antecedent $\pi(B_1)\ \ldots\ \pi(B_n)$ is replaced by $\top$ (corresponding to the empty body). □

Each derivation in a proof is a particular application of Modus Ponens where, once the body (conjunction of literals B) of a rule R ($p \leftarrow B$) has been proved, then the head p can be concluded.

$$\dfrac{\dfrac{\dfrac{\top}{\mathit{drive}}\,(d) \qquad \dfrac{\top}{\mathit{drunk}}\,(k)}{\mathit{punish}}\,(l)}{\mathit{prison}}\,(e) \qquad\qquad \dfrac{\dfrac{\dfrac{\top}{\mathit{resist}}\,(r)}{\mathit{punish}}\,(m)}{\mathit{prison}}\,(e) \qquad\qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\top}{\mathit{drive}}\,(d) \qquad \dfrac{\top}{\mathit{drunk}}\,(k)}{\mathit{punish}}\,(l)}{\mathit{sentence}}\,(s)}{\mathit{punish}}\,(n)}{\mathit{prison}}\,(e)$$

Fig. 3. Some proofs for atom *prison* (the rightmost proof is redundant).

Example 5 (*Ex. 1 continued*)
Program P_1 is positive and, in fact, completely labelled, so we can identify each rule with its label. Atom *prison* can be derived in P_1 using the two proofs on the left in Figure 3. These two proofs have a clear correspondence to causes $(d * k) \cdot le$ and rme depicted in Figure 1(b). In fact, the least model I of P_1 assigns causal value $I(\mathit{punish}) = (d * k) \cdot le + rme$. □

Let P be a positive, completely labelled program. Given a proof π, we define its graph G_π as follows. For each sub-derivation in π of the form $\pi(p)$ in Definition 8 we include an edge (l_i, m) where m is the label of rule R and l_i is the label of the top-level rule in $\pi(B_i)$, for all $i = 1, \ldots, n$. The vertices in G_π exclusively collect the labels in those edges. We define $\mathit{cause}(\pi) \overset{\text{def}}{=} G_\pi^*$. The two left proofs in Figure 3 are then obviously mapped to the causes in Figure 1(b). If Π is a set of proofs, we define $\mathit{causes}(\Pi) \overset{\text{def}}{=} \{\mathit{cause}(\pi) \mid \pi \in \Pi\}$.

A proof can be sometimes redundant, in the sense that some of its derivation steps could be removed. A natural way of defining a non-redundant proof is resorting to its associated graph. We say that a proof $\pi(p)$ for an atom p in a positive, completely labelled program P is *redundant* if there exists another proof for p, $\pi'(p)$, such that $\mathit{cause}(\pi(p)) \leq \mathit{cause}(\pi'(p))$, in other words, we can build another proof π' with a smaller associated graph.

Example 6
Suppose that we introduce an atom *sentence* which acts as a synonym for *punish*. Furthermore, assume law m mentions *sentence* as its head now, instead of *punish*. Hence, let P_2 be program:

$$\begin{array}{rlrlrl} l: & punish \leftarrow drive, drunk & d: & drive & n: & punish \leftarrow sentence \\ m: & sentence \leftarrow resist & k: & drunk & s: & sentence \leftarrow punish \\ e: & prison \leftarrow punish & r: & resist & & \end{array}$$

Then, the rightmost proof shown in Figure 3 together with its associated cause $(d * k) \cdot lsne$ is redundant, since the (still valid) leftmost proof in Figure 3 for *prison* has an associated stronger cause (or smaller graph) $(d * k) \cdot le$. Considering the positive loop formed by n and s, one may wonder why it does not spoil the computation of T_{P_2} to iterate forever (adding more and more concatenations of n and s). The reason is that, at a given point, subsequent iterations yield redundant causes subsumed by previous steps. In particular, the iteration of T_{P_2} yields the steps:

i	*drive*	*drunk*	*resist*	*sentence*	*punish*	*prison*
1	d	k	r	0	0	0
2	d	k	r	rm	$(d*k)\cdot l$	0
3	d	k	r	$rm + (d*k)\cdot ls$	$(d*k)\cdot l + rmn$	$(d*k)\cdot le$
4	d	k	r	$rm + (d*k)\cdot ls$	$(d*k)\cdot l + rmn$	$((d*k)\cdot l + rmn)\cdot e$

reaching a fixpoint at step 4. The value for *sentence* at step 4 would actually be the sum of rm (derived from *resist*) with the value of *punish* in the previous step, $(d * k) \cdot l + rmn$ followed by s. This corresponds to:

$$\begin{array}{rll} rm + (\underbrace{(d*k)\cdot l + rmn}_{punish})\cdot s & = rm + (d*k)\cdot ls + rmns & \text{distributivity} \\ & = rm + rmns + (d*k)\cdot ls & \text{commutativity} \\ & = rm + rm\cdot ns + (d*k)\cdot ls & \text{associativity} \\ & = rm + 1\cdot rm\cdot ns + (d*k)\cdot ls & \text{identity} \\ & = rm + (d*k)\cdot ls & \text{absorption for `$+$' and `$\cdot$'} \end{array}$$

That is, iterating the loop *rmns* is redundant since a stronger cause *rm* was obtained before. □

Theorem 3
Let P be a positive, completely labelled program, and Π_p the set of non-redundant proofs for some atom p with respect to P. If I denotes the least model of P, then:

$$G \in causes(\Pi_p) \quad \text{iff} \quad G \text{ is a maximal cause in } I(p) \qquad \square$$

Completely labelled programs are interesting for establishing the correspondence in the theorem above, but there are several scenarios in which one may be interested in disregarding the effect of rules in a program or in identifying a group of rules under the same label.

Example 7
Let P_3 be the following variation of P_2:

$$\begin{array}{rlrll} z: & sentence \leftarrow drive, drunk & d: & drive & punish \leftarrow sentence \\ z: & punish \leftarrow resist & k: & drunk & sentence \leftarrow punish \\ e: & prison \leftarrow punish & r: & resist & \end{array}$$

where l and m in P_2 are now just two cases of a common law z, and *punish* and *sentence* depend on each other through unlabelled rules. □

Removing the labels in the positive cycle between *sentence* and *punish* captures the idea that, since they are synonyms, whenever we have a cause for *sentence*, it immediately becomes a cause for *punish* and vice versa. By iterating the T_P operator, it is not difficult to see that the least causal model I_3 makes the assignments $I_3(sentence) = I_3(punish) = (d*k)\cdot z + rz$ (that is *sentence* and *punish* are equivalent) and $I_3(prison) = (d*k)\cdot ze + rze$. This result could also be computed from the least model I_2 for P_2 by replacing l and m by z and "removing" n and s (that is, replacing them by 1). This is, in fact, a general property we formalise as follows. Given two causal terms t, u and a label l, we define $t[l \mapsto u]$ as the result of replacing label l in t by term u.

Theorem 4
Let P be a positive causal logic program and P' be the result of replacing a label l in P by some u, where u is any label or 1. Furthermore, let I and I' be the least models of P and P', respectively. Then, $I'(p) = I(p)[l \mapsto u]$ for any atom p. □

In particular, in our example, $I_3(p) = I_2(p)[l \mapsto z][m \mapsto z][n \mapsto 1][s \mapsto 1]$, for any atom p. If we remove all labels in a program, we eventually get a standard, unlabelled program. Obviously, its least model will be classical, since removing all labels in causal terms, eventually collapses all of them to $\{0,1\}$. As a result, we can easily establish the following correspondence.

Theorem 5
Let P be a causal positive logic program and P' its unlabelled version. Furthermore, let I be the least causal model of P and I' the least classical model of P'. Then $I' = I^{cl}$. □

5 Default negation

To introduce default negation, let us consider the following variation of our running example.

Example 8
Assume now that law e is a default and that there may be exceptional cases in which punishment is not effective. In particular, some of such exceptions are a pardon, that the punishment was revoked, or that the person has diplomatic immunity. A possible program P_4 encoding this variant of the scenario is:

$$\begin{array}{rlrll}
l: & punish \leftarrow drive, drunk & d: & drive & abnormal \leftarrow pardon \\
m: & punish \leftarrow resist & k: & drunk & abnormal \leftarrow revoke \\
e: & prison \leftarrow punish, not\ abnormal & r: & resist & abnormal \leftarrow diplomatic
\end{array}$$

This program has a unique stable model which still keeps *prison* true, since *no proof for abnormal* could be obtained, i.e., no exception occurred. □

From a causal perspective, saying that the lack of an exception is part of a cause (e.g., for imprisonment) is rather counterintuitive. It is not the case that we go to prison because of not receiving a pardon, not having a punishment revocation, not being a diplomatic, or whatever possible exception that might be added in the future[2]. Instead, as nothing violated default e, the justifications for *prison* should be those shown in Figure 1(a). In this way, falsity becomes the *default situation*

[2] A case of the well-known *qualification problem* (McCarthy 1977), i.e., the impossibility of listing all the possible conditions that prevent an action to cause a given effect. Appendix B contains a more elaborated example showing how the qualification problem may affect causal explanations when inertia is involved.

that is broken when a cause is found[3]. This interpretation carries over to negative literals, so that the presence of *not p* in a rule body does not propagate causal information, but instead is a check for the absence of an exception. To capture this behaviour, we proceed to extend the traditional program reduct (Gelfond and Lifschitz 1988) to causal logic programs.

Definition 9 (*Program reduct*)
The *reduct* of program P with respect to causal interpretation I, in symbols P^I, is the result of:

1. removing from P all rules R, s.t. $I(B) \neq 0$ for some negative literal $B \in body(R)$;
2. removing all negative literals from the remaining rules of P. □

An interpretation I is a *causal stable model* of program P iff I is the least causal model of P^I.

Example 9 (*Ex. 8 continued*)
Suppose that we add atoms $(p : pardon)$ and $(d : diplomatic)$ to program P_4. The only stable model I of this extended program makes $I(prison) = 0$ and $I(abnormal) = p + d$ as expected. □

Theorem 6 (*Correspondence to non-causal stable models*)
Let P be a causal logic program and P' its unlabelled version. Then:

1. If I is a causal stable model of P, then I^{cl} is a stable model of P'.
2. If I' is a stable model of P' then there is a unique causal stable model I of P s.t. $I' = I^{cl}$. □

This theorem also shows a possible method for computing causal stable models of a program P. We may first run a standard ASP solver on the unlabelled version of P to obtain a stable model I'. This stable model I' has a corresponding causal stable model I, such that $I' = I^{cl}$ and both interpretations coincide in their assignment of 0's. Therefore, $P^I = P^{I'}$ and we can use the latter to iterate the T_P operator and obtain the least causal model of this reduct, which will mandatorily be a causal stable model due to Theorem 6.

6 Related Work

(Cabalar 2011) already introduced the main motivations of our work, but used *ad hoc* operations on proof trees without resorting to algebraic structures. A preliminary version (Cabalar and Fandinno 2013) of the current approach relied on chains of labels but was actually *weaker*, missing basic properties we can derive now from causal graphs.

There exists a vast literature on causal reasoning in Artificial Intelligence. Papers on reasoning about actions and change (Lin 1995; McCain and Turner 1997; Thielscher 1997) have been traditionally focused on using causal inference to solve representational problems (mostly, the frame, ramification and qualification problems) without paying much attention to the derivation of cause-effect relations. Perhaps the most established AI approach for causality is relying on *causal networks* (Pearl 2000; Halpern and Pearl 2005; Halpern 2008). In this approach, it is possible to conclude cause-effect relations like "A has caused B" from the behaviour of structural equations by applying the counterfactual interpretation from Hume (1748): "had A not happened, B would not have happened." As discussed by Hall (2004), the counterfactual-based definition of causation corresponds to recognising some kind of *dependence* relation in the behaviour of a non-causal system description. As opposed to this, Hall considers a different (and incompatible)

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

definition where causes must be connected to their effects via *sequences of causal intermediates*, something that is closer to our explanations in terms of causal graphs.

Apart from the different AI approaches and attitudes towards causality, from the technical point of view, the current approach can be classified as a *labelled deductive system* (Broda et al. 2004). In particular, the work that has had a clearest and most influential relation to the current proposal is the *Logic of Proofs* (**LP**) by Artëmov (2001). We have borrowed from that formalism part of the notation for our causal terms and rule labellings and the fundamental idea of keeping track of justifications by considering rule applications.

Focusing on LP, our work obviously relates to explanations as provided by approaches to debugging in ASP (Gebser et al. 2008; Pontelli et al. 2009; Schulz et al. 2013; Damásio et al. 2013). These works aim at explaining discrepancies between an expected result and the obtained stable models. Accordingly, explanations usually contain all possible ways to derive an atom or to prevent its derivation, including paths through negation. This differs from a KR orientation where only the cause-effect relations that "break the norm" should be considered relevant. This point of view is also shared, e.g., by the counterfactual-based causal LP approach (Vennekens 2011).

A more far-fetched resemblance exists to work on the analysis of tabled Prolog computations. There, the goal is to identify potential causes for non-termination of program evaluations, which can be achieved examining so-called *forest logs*, i.e., a log of table operations for a computation. By adding unique labels for rules (with the original intention to disambiguate analysis results, cf. Liang and Kifer (2013), however not as an explicit means for representing knowledge), in principle a forest log implicitly contains the information necessary to read of the causal model of a completely labelled positive causal logic program.

7 Conclusions

In this paper we have provided a multi-valued semantics for normal logic programs whose truth values form a lattice of causes. A cause is nothing else but a graph of rule labels that reflects some order of rule applications. In this way, a model assigns to each true atom a value that contains justifications for its derivation from the existing rules. We have further provided three basic operations on the lattice: an addition, that stands for alternative, independent justifications; a product, that represents joint interaction of causes; and a concatenation that reflects rule application. We have shown that, for positive programs, there exists a least model that coincides with the least fixpoint of a direct consequences operator, analogous to van Emden and Kowalski (1976). With this, we are able to prove a direct correspondence between the semantic values we obtain and the syntactic idea of proof. These results have been extrapolated to stable models of programs with default negation, understanding the latter as "absence of cause."

Several topics remain open for future study. An interesting issue is to replace the syntactic definition by a reduct in favour of a logical treatment of default negation, as has been done for (non-causal) stable models and their characterisation in terms of Equilibrium Logic (Pearce 2006). Regarding the representation of causal information, a natural next step would be the consideration of syntactic operators for more specific knowledge like the influence of a particular event or label in a conclusion, expressing necessary or sufficient causes, or even dealing with counterfactuals. Further ongoing work is focused on implementation, complexity assessment, and an extension to disjunctive programs. Exploring related areas of KR and reasoning, such as, e.g., Paraconsistent Reasoning and Belief Revision, seems promising with respect to extending the range of problems to which our approach may effectively be applied.

References

ARTËMOV, S. N. 2001. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic 7,* 1, 1–36.

BREWKA, G., EITER, T., AND TRUSZCZYNSKI, M. 2011. Answer set programming at a glance. *Commun. ACM 54,* 12, 92–103.

BRODA, K., GABBAY, D., LAMB, L., AND RUSSO., A. 2004. *Compiled Labelled Deductive Systems: A Uniform Presentation of Non-Classical Logics*. Research Studies Press.

CABALAR, P. 2011. Logic programs and causal proofs. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*. AAAI.

CABALAR, P. AND FANDINNO, J. 2013. An algebra of causal chains. In *Proc. of the 6th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP'13).*

DAMÁSIO, C. V., ANALYTI, A., AND ANTONIOU, G. 2013. Justifications for logic programming. In *Proc. of the 12th Intl. Conf. on Logic Programming and Nonmonotonic Reasoning, (LPNMR'13).* Lecture Notes in Computer Science, vol. 8148. Springer, 530–542.

GEBSER, M., PÜHRER, J., SCHAUB, T., AND TOMPITS, H. 2008. Meta-programming technique for debugging answer-set programs. In *Proc. of the 23rd Conf. on Artificial Inteligence (AAAI'08).* 448–453.

GELFOND, M. AND LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In *Logic Programming: Proc. of the Fifth International Conference and Symposium (Volume 2)*, R. A. Kowalski and K. A. Bowen, Eds. MIT Press, Cambridge, MA, 1070–1080.

HALL, N. 2004. *Two concepts of causality*. 181–276.

HALPERN, J. Y. 2008. Defaults and normality in causal structures. In *Proc. of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2008).* 198–208.

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.

HUME, D. 1748. An enquiry concerning human understanding. Reprinted by Open Court Press, LaSalle, IL, 1958.

KIFER, M. AND SUBRAHMANIAN, V. S. 1992. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming 12*.

LIANG, S. AND KIFER, M. 2013. A practical analysis of non-termination in large logic programs. *TPLP 13,* 4-5, 705–719.

LIN, F. 1995. Embracing causality in specifying the indirect effects of actions. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, C. S. Mellish, Ed. Morgan Kaufmann, Montreal, Canada.

MCCAIN, N. AND TURNER, H. 1997. Causal theories of action and change. In *Proc. of the AAAI-97.* 460–465.

MCCARTHY, J. 1977. Epistemological problems of Artificial Intelligence. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI).* MIT Press, Cambridge, MA, 1038–1044.

MCCARTHY, J. 1998. Elaboration tolerance. In *Proc. of the 4th Symposium on Logical Formalizations of Commonsense Reasoning (Common Sense 98).* London, UK, 198–217. Updated version at `http://www-formal.stanford.edu/jmc/elaboration.ps`.

PEARCE, D. 2006. Equilibrium logic. *Ann. Math. Artif. Intell. 47,* 1-2, 3–41.

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

PONTELLI, E., SON, T. C., AND EL-KHATIB, O. 2009. Justifications for logic programs under answer set semantics. *Theory and Practice of Logic Programming 9,* 1, 1–56.

SCHULZ, C., SERGOT, M., AND TONI, F. 2013. Argumentation-based answer set justification. In *Proc. of the 11th Intl. Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense'13).*

STUMME, G. 1997. Free distributive completions of partial complete lattices. *Order 14*, 179–189.

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

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

Vennekens, J. 2011. Actual causation in cp-logic. *TPLP 11,* 4-5, 647–662.

Appendix A. Auxiliary figures

$$\begin{array}{ccc|ccc|ccc}
\multicolumn{3}{c|}{\textit{Associativity}} & \multicolumn{3}{c|}{\textit{Commutativity}} & \multicolumn{3}{c}{\textit{Absorption}} \\
\hline
t+(u{+}w) & = & (t{+}u)+w & t+u & = & u+t & t & = & t+(t*u) \\
t*(u{*}w) & = & (t{*}u)*w & t*u & = & u*t & t & = & t*(t{+}u)
\end{array}$$

$$\begin{array}{ccc|ccc|ccc|ccc}
\multicolumn{3}{c|}{\textit{Distributive}} & \multicolumn{3}{c|}{\textit{Identity}} & \multicolumn{3}{c|}{\textit{Idempotence}} & \multicolumn{3}{c}{\textit{Annihilator}} \\
\hline
t+(u{*}w) & = & (t{+}u)*(t{+}w) & t & = & t+0 & t & = & t+t & 1 & = & 1+t \\
t*(u{+}w) & = & (t{*}u)+(t{*}w) & t & = & t*1 & t & = & t*t & 0 & = & 0*t
\end{array}$$

Fig. 4. Sum and product satisfy the properties of a completely distributive lattice.

Appendix B. An example of causal action theory

In this section we consider a more elaborated example from Pearl (2000).

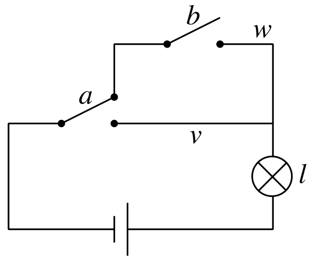
Example 10
Consider the circuit in Figure 5 with two switches, a and b, and a lamp l. Note that a is the main switch, while b only affects the lamp when a is up. Additionally, when the light is on, we want to track which wire section, v or w, is conducting current to the lamp. $\square$

As commented by Pearl (2000), the interesting feature of this circuit is that, seen from outside as a black box, it behaves exactly as a pair of independent, parallel switches, so it is impossible to detect the causal dependence between a and b by a mere observation of performed actions and their effects on the lamp. Figure 5 also includes a possible representation for this scenario; let us call it program P_5. It uses a pair of fluents $up(X)$ and $down(X)$ for the position of switch X, as well as *on* and *off* to represent the state of the lamp. Fluents $up(X)$ and $down(X)$ (respectively, *on* and *off*) can be seen as the strong negation of each other, although we do not use an operator for that purpose[4]. Action $m(X,D)$ stands for "move switch X in direction $D \in \{u,d\}$" (*up* and *down*, respectively). Actions between state t and $t+1$ are located in the resulting state. Finally, we have also labelled inertia laws (by i) to help keeping track of fluent justifications inherited by persistence.

Suppose we perform the following sequence of actions: we first move down both switches, next switch b is moved first up and then down, and finally we move up switch a. Assume also that each action occurrence is labelled with the action name so that, for instance, moving b up in Situation 1 corresponds to the program fact $m(b,u)_1 : m(b,u)_1$. The table in Figure 6 shows the resulting temporal projection. Note how the lamp turns on in Situation 1 but only because of v, that is, moving a down. Movements of b at 2 and 3 do not affect the lamp, and its causal explanation ($down(a)$) is maintained by inertia. In Situation 4, the lamp is still on but the reason has changed. The explanation this time is that we had closed down b at 3 (and this persisted by inertia) while we have just moved a up, firing rule w.

[4] Notice how strong negation would point out the cause(s) for a boolean fluent to take value false, whereas default negation represents the absence of cause.

Circuit diagram



Inertia

$$
\begin{array}{lrcl}
i: & up(X)_{t+1} & \leftarrow & up(X)_t,\ not\ down(X)_{t+1} \\
i: & down(X)_{t+1} & \leftarrow & down(X)_t,\ not\ up(X)_{t+1}
\end{array}
$$

Direct effects

$$
\begin{array}{rcl}
up(X)_t & \leftarrow & m(X,u)_t \\
down(X)_t & \leftarrow & m(X,d)_t
\end{array}
$$

Causal rules (indirect effects)

$$
\begin{array}{lrcl}
v: & on_t & \leftarrow & down(a)_t \\
w: & on_t & \leftarrow & up(a)_t,\ down(b)_t \\
 & \mathit{off}_t & \leftarrow & up(a)_t,\ up(b)_t
\end{array}
$$

Constraints

$$
\begin{array}{rcl}
\bot & \leftarrow & m(X,u)_t,\ m(X,d)_t \\
\bot & \leftarrow & up(X)_t,\ down(X)_t \\
\bot & \leftarrow & on_t,\ \mathit{off}_t
\end{array}
$$

Initial state

$up(a)_0 \quad up(b)_0 \quad \mathit{off}_0$

Fig. 5. A circuit with two switches together with a possible representation.

t	0	1	2	3	4
Actions		$m(a,d)_1,\ m(b,d)_1$	$m(b,u)_2$	$m(b,d)_3$	$m(a,u)_4$
$up(a)_t$	1	0	0	0	$m(a,u)_4$
$down(a)_t$	0	$m(a,d)_1$	$m(a,d)_1 \cdot i$	$m(a,d)_1 \cdot i$	0
$up(b)_t$	1	0	$m(b,u)_2$	0	0
$down(b)_t$	0	$m(b,d)_1$	0	$m(b,d)_3$	$m(b,d)_3 \cdot i$
on_t	0	$m(a,d)_1 \cdot v$	$m(a,d)_1 \cdot iv$	$m(a,d)_1 \cdot iv$	$(\ m(b,d)_3 \cdot i * m(a,u)_4\ ) \cdot w$
off_t	1	0	0	0	0

Fig. 6. Temporal projection of a sequence of actions for program P_5.

This example also illustrates why we are not interested in providing negative justifications through default negation. This would mean to explicitly include non-occurrences of actions that might otherwise have violated inertia. For instance, the explanation for on_2 would include the fact that we did not perform $m(a,u)_2$. Including this information for one transition is perhaps not so cumbersome, but suppose that, from 2 we executed a high number of transitions without performing any action. The explanation for on_3 would *additionally* collect that we did not perform $m(a,u)_3$ either. The explanation for on_4 should also collect the negation of further possibilities: moving a up at 4; three movements of a up, down and up; moving b at 3 and both switches at 4; moving both switches at 3 and b at 4; etc. It is easy to see that negative explanations grow exponentially: at step t we would get the negation of *all possible plans* for making on_t false, while indeed, *nothing has actually happened* (everything persisted by inertia).

Appendix C. Proofs

In order to improve clarity, for any cause $G = \langle V, E \rangle$ we use the notation $V(G)$ and $E(G)$ to refer to V and E respectively.

Proposition 1 (*Monotonicity*)
Let G, G' be a pair of causes with $G \leq G'$. Then, for any cause H:

$$G * H \leq G' * H, \qquad G \cdot H \leq G' \cdot H \qquad \text{and} \qquad H \cdot G \leq H \cdot G'$$

Proof. First we will show that $G * H \leq G' * H$. Suppose that $E(G * H) \not\supseteq E(G' * H)$ and let (l_1, l_2) be an edge in $E(G' * H)$ but not in $E(G * H')$, i.e. $(l_1, l_2) \in E(G' * H) \backslash E(G * H')$.

Thus, since by product definition $E(G' * H) = E(G') \cup E(H)$, it follows that either $(l_1, l_2) \in E(G')$ or $(l_1, l_2) \in E(H)$. It is clear that if $(l_1, l_2) \in E(H)$ then $(l_1, l_2) \in E(G * H) = E(G) \cup E(H)$. Furthermore, since $G \leq G'$ it follows that $E(G) \supseteq E(G')$, if $(l_1, l_2) \in E(G')$ then $(l_1, l_2) \in E(G)$ and consequently $(l_1, l_2) \in E(G * H) = E(G) \cup E(H)$. That is $E(G * H) \supseteq E(G' * H)$ and then $G * H \leq G' * H$. Note that $V(G * H) \supseteq V(G' * H)$ follows directly from $E(G * H) \supseteq E(G' * H)$ and the fact that every vertex has and edge to itself.

To show that $G \cdot H \leq G' \cdot H$ (the case for $H \cdot G \leq H \cdot G'$ is analogous) we have has to show that, in addition to the previous, for every edge $(l_G, l_H) \in E(G' \cdot H)$ with $l_G \in V(G')$ and $l_H \in V(H)$ it holds that $(l_G, l_H) \in E(G \cdot H)$. Simply note that since $G \leq G'$ it follows $V(G) \supseteq V(G)'$ and then $l_G \in V(G)$. Consequently $(l_G, l_H) \in E(G \cdot H)$. □

Proposition 2 (*Distributivity*)
For every pair of sets of causal graphs S and S', it holds that

$$\left(\prod S\right) \cdot \left(\prod S'\right) = \prod \left\{\ G \cdot G' \ \middle|\ G \in S \text{ and } G' \in S'\ \right\}.$$

Proof. For readability sake, we define two causes

$$G_R \stackrel{\text{def}}{=} \left(\prod S\right) \cdot \left(\prod S'\right) \qquad \text{and} \qquad G_L \stackrel{\text{def}}{=} \prod \left\{\ G \cdot G' \ \middle|\ G \in S \text{ and } G' \in S'\ \right\}$$

and we assume that both S and S' are not empty sets. Note that $\prod \emptyset = \mathbf{C}_{Lb} = \prod \{G_\emptyset\}$. Then, by product definition, it follows that

$$E(G_L) = \left(\bigcup \left\{\ E(G) \ \middle|\ G \in S\ \right\} \cup \bigcup \left\{\ E(G') \ \middle|\ G \in S'\ \right\} \cup E_L\right)^*$$
$$E(G_R) = \left(\bigcup \left\{\ E(G) \cup E(G') \cup E_R(G, G') \ \middle|\ G \in S \text{ and } G' \in S'\ \right\}\right)^*$$

where

$$E_L \quad = \left\{\ (l, l') \ \middle|\ l \in \bigcup \left\{\ V(G) \ \middle|\ G \in S\ \right\} \text{ and } l' \in \bigcup \left\{\ V(G') \ \middle|\ G' \in S'\ \right\}\ \right\}$$
$$E_R(G, G') = \left\{\ (l, l') \ \middle|\ l \in V(G) \text{ and } l' \in V(G')\ \right\}$$

Furthermore let $E_R = \bigcup \left\{\ E_R(G, G') \ \middle|\ G \in S \text{ and } G' \in S'\ \right\}$. For every edge $(l, l') \in E_L$ there are a pair of causes $G \in S$ and $G' \in S'$ s.t. $l \in V(G)$ and $l' \in V(G')$ and then $(l, l') \in E_R(G, G')$ and so $(l, l') \in E_R$. Moreover, for every edge $(l, l') \in E_R$ there are a pair of causes $G \in S$ and $G' \in S'$ s.t. $(l, l') \in E_R(G, G')$ with $l \in V(G)$ and $l' \in V(G)'$. So that $(l, l') \in E_L$. That is $E_L = E_R$. Then

$$E(G_R) = \left(\bigcup \left\{\ E(G) \ \middle|\ G \in S\ \right\} \cup \bigcup \left\{\ E(G') \ \middle|\ G' \in S'\ \right\} \cup E_R\right)^*$$
$$= \left(E(G_L) \backslash E_L \cup E_R\right)^* = \left(E(G_L)\right)^* = E(G_L)$$

Consequently $G_L = G_R$. □

Proposition 3
For every cause $G = \langle V, E \rangle$ it holds that $G = \prod \{ \ l \cdot l' \mid (l,l') \in E \ \}$.

Proof. Let G' be a cause s.t. $G' = \prod \{ \ l \cdot l' \mid (l,l') \in E \ \}$. Then for every edge $(l,l') \in E(G)$ it holds that $(l,l') \in E(l \cdot l')$ and then $(l,l') \in E(G') = \bigcup \{ \ E(l \cdot l') \mid (l,l') \in E \ \}$, i.e. $E(G) \subseteq E(G')$. Furthermore for every $(l,l') \in E(G')$ there is $l_i \cdot l_j$ s.t. $(l,l') \in l_i \cdot l_j$ and $(l_i,l_j) \in E(G)$. Then, since $E(l_i \cdot l_j) = \{(l_i,l_j)\}$ it follows that $(l,l') \in E(G)$, i.e. $E(G) \supseteq E(G')$. Consequently $G = G' = \prod \{ \ l \cdot l' \mid (l,l') \in E \ \}$. □

Proposition 4 (*Infimum*)
Any set of causes S has a $\leq$-infimum given by their product $\prod S$.

Proof. By definition $\prod S$ is the causal graph of which vertex and edges are respectively the sets $V(\prod S) = \bigcup \{ \ V(G) \mid G \in S \ \}$ and $E(\prod S) = \bigcup \{ \ E(G) \mid G \in S \ \}$. It is easy to see that $\prod S$ is the supremum of the subgraph relation, so that, since for every pair of causes $G \leq G'$ iff $G \supseteq G'$, it follows that infimum of S w.r.t. $\leq$. □

Proof of Theorem 1. Let $\mathbf{F}$ be the set of filters over the lower semilattice $\langle \mathbf{C}_{Lb}, * \rangle$. Stumme was showed in (Stumme 1997) that the concept lattice $\underline{\mathfrak{B}}\langle \mathbf{F}, \mathbf{V}_{Lb}, \Delta \rangle$ (with $F \Delta I \Leftrightarrow F \cap I \neq \emptyset$) is isomorphic to the free completely distributive complete lattice generated by the partial lattice $\langle \mathbf{C}_{Lb}, +, * \rangle$ where $+$ and $*$ are two partial functions corresponding with the supremum and infimum. Note that, in our particular, case for every set of causes S its infimum is defined and is $\prod S$ but the supremum is not. Thus $\mathbf{V}_{Lb}$ is the set of ideals over the partial lattice $\langle \mathbf{C}_{Lb}, +, * \rangle$, i.e. every $I \in \mathbf{V}_{Lb}$ is also closed under defined suprema (since it never is defined). He also show that the elements of such lattice are described as pairs

$$\{ \ (\mathbf{F}_t, \mathbf{I}_t) \mid \mathbf{F}_t \subseteq \mathbf{F}, \ \mathbf{I}_t \subseteq \mathbf{V}_{Lb}, \ \mathbf{F}_t^I = \mathbf{I}_t \text{ and } \mathbf{F}_t = \mathbf{I}_t^I \ \}$$

where

$$\mathbf{F}_t^I = \{ \ I \in \mathbf{I} \mid \forall F \in \mathbf{F}_t : F \cap I \neq \emptyset \ \}$$
$$\mathbf{I}_t^I = \{ \ F \in \mathbf{F} \mid \forall I \in \mathbf{I}_t : F \cap I \neq \emptyset \ \}$$

That is, every element is in the form $\langle \mathbf{I}_t^I, \mathbf{I}_t \rangle$. Furthermore infima and suprema can be described as follows:

$$\bigwedge_{t \in T} (\mathbf{I}_t^I, \mathbf{I}_t) = \Big(\bigcap_{t \in T} \mathbf{I}_t^I, \Big(\bigcup_{t \in T} \mathbf{I}_t \Big)^{II} \Big)$$
$$\bigvee_{t \in T} (\mathbf{I}_t^I, \mathbf{I}_t) = \Big(\Big(\bigcup_{t \in T} \mathbf{I}_t^I \Big)^{II}, \ \bigcap_{t \in T} \mathbf{I}_t \ \Big)$$

We will show that $\varepsilon_I : \underline{\mathfrak{B}} \longrightarrow \mathbf{V}_{Lb}$ given by $(\mathbf{I}_t^I, \mathbf{I}_t) \mapsto \bigcap \mathbf{I}_t$ is an isomorphism between $\langle \underline{\mathfrak{B}}, \vee, \wedge \rangle$ and $\langle \mathbf{V}_{Lb}, \cup, \cap \rangle$. Note that, since $\prod \emptyset = G_\emptyset$ it holds that the empty set is not close under defined infimum and then it is not a filter, i.e. $\emptyset \notin \mathbf{F}$, and then for every filter $F \in \mathbf{F}$ it holds that $G_\emptyset \in F$. Thus if $\mathbf{I}_t = \emptyset$ follows that $\mathbf{I}_t^I = \mathbf{F}$ and then $\mathbf{I}_t^{II} = \{ I \in \mathbf{V}_{Lb} \mid G_\emptyset \in I \} = \mathbf{C}_{Lb} \neq \mathbf{I}_t$. That is, $\langle \emptyset^I, \emptyset \rangle \notin \underline{\mathfrak{B}}$.

We will show that for every ideal $I_t \in \mathbf{I}$ and for every set of ideals $\mathbf{I}_t \subseteq \mathbf{I}$ s.t. $I_t = \bigcap \mathbf{I}_t$ it holds that

$$(\mathbf{I}_t^I, \mathbf{I}_t) \in \underline{\mathfrak{B}}\langle \mathbf{F}, \mathbf{I}, \Delta \rangle \Longleftrightarrow \mathbf{I}_t = \{ \ I \in \mathbf{I} \mid I_t \subseteq I \ \} \qquad (2)$$

and consequently ε_I is a bijection between and $\underline{\mathfrak{B}}$ and $\mathbf{V}_{Lb}$.

Suppose that $(\mathbf{I}_t^I, \mathbf{I}_t) \in \underline{\mathfrak{B}}\langle \mathbf{F}, \mathbf{I}, \Delta \rangle$. For every $I \in \mathbf{I}_t$ it holds that $I_t \subseteq I$. So suppose there is $I \in \mathbf{I}$ s.t. $I_t \subseteq I$ and $I \notin \mathbf{I}_t$. Then there is $F \in \mathbf{I}_t^I$ s.t. $I \cap F = \emptyset$ and for every element $I' \in \mathbf{I}_t$ it holds that $I' \cap F \neq \emptyset$. Pick a causal graph G s.t. $G = \prod\{G' \mid G' \in I' \cap F \text{ and } I' \in \mathbf{I}_t\}$. Since for every G' it holds $G' \in F$ and $G \leq G'$ follows that $G \in F$ (F is close under infimum) and $G \in I'$ (every I' is close under $\leq$). That is, for every $I' \in \mathbf{I}_t$ it holds that $G \in I' \cap F$ and then, since $I_t = \bigcap \mathbf{I}_t$, it also holds that $G \in I_t \cap F$ and since $I_t \subseteq I$ also $G \in I \cap F$ which contradict that $I \cap F = \emptyset$. So that $I \in \mathbf{I}_t$ and it holds that

$$(\mathbf{I}_t^I, \mathbf{I}_t) \in \underline{\mathfrak{B}}\langle \mathbf{F}, \mathbf{I}, \Delta \rangle \Longrightarrow \mathbf{I}_t = \left\{ I \in \mathbf{I} \;\middle|\; I_t \subseteq I \right\}$$

Suppose that $\mathbf{I}_t = \{I \in \mathbf{I} \mid I_t \subseteq I\}$ but $(\mathbf{I}_t^I, \mathbf{I}_t) \in \underline{\mathfrak{B}}\langle \mathbf{F}, \mathbf{I}, \Delta \rangle$, i.e. $\mathbf{I}_t \neq \mathbf{I}_t^{II}$. Note that $\mathbf{I}_t \subseteq \mathbf{I}_t^{II}$ because otherwise there are $I \in \mathbf{I}_t$ and $F \in \mathbf{I}_t^I$ s.t. $I \cap F = \emptyset$ which is a contradiction with the fact that for every $F \in \mathbf{I}_t^I$ and $I \in \mathbf{I}_t$ it holds that $F \cap I \neq \emptyset$.

So, there is $I \in \mathbf{I}_t^{II}$ s.t. $I \notin \mathbf{I}_t$, i.e. for every $F \in \mathbf{I}_t^I$ it holds that $F \cap I \neq \emptyset$ but $I_t \not\subseteq I$. Pick $G \in I_t \backslash I$ and $F = \{G' \mid G \leq G'\}$. It is clear that $F \in \mathbf{F}$ and $F \cap I_t \neq \emptyset$ because $G \in I_t$, so that $F \in \mathbf{I}_t^I$. Furthermore $F \cap I = \emptyset$, because $G \notin I$, which is a contradiction with the assumption. Thus

$$(\mathbf{I}_t^I, \mathbf{I}_t) \in \underline{\mathfrak{B}}\langle \mathbf{F}, \mathbf{I}, \Delta \rangle \Longleftarrow \mathbf{I}_t = \left\{ I \in \mathbf{I} \;\middle|\; I_t \subseteq I \right\}$$

Now, we will show that $(\mathbf{I}_1^I, \mathbf{I}_1) \vee (\mathbf{I}_2^I, \mathbf{I}_2) = (\mathbf{I}_3^I, \mathbf{I}_3)$ iff $I_1 \cup I_2 = I_3$. From the above statement follows that

$$\begin{aligned}
\mathbf{I}_1 \cap \mathbf{I}_2 &= \{I \in \mathbf{I} \mid I_1 \subseteq I \text{ and } I_2 \subseteq I\} = \\
&= \{I \in \mathbf{I} \mid I_1 \cup I_2 \subseteq I\} \\
\mathbf{I}_3 &= \{I \in \mathbf{I} \mid I_3 \subseteq I\}
\end{aligned}$$

That is, $\mathbf{I}_1 \cap \mathbf{I}_2 = \mathbf{I}_3$ iff $I_1 \cup I_2 = I_3$ and by definition of $\vee$ the first is equivalent to $(\mathbf{I}_1^I, \mathbf{I}_1) \vee (\mathbf{I}_2^I, \mathbf{I}_2) = (\mathbf{I}_3^I, \mathbf{I}_3)$.

Finally we will show that $(\mathbf{I}_1^I, \mathbf{I}_1) \wedge (\mathbf{I}_2^I, \mathbf{I}_2) = (\mathbf{I}_3^I, \mathbf{I}_3)$ iff $I_1 \cap I_2 = I_3$. It holds that

$$\begin{aligned}
(\mathbf{I}_1 \cup \mathbf{I}_2)^{II} &= \Big(\left\{ I \in \mathbf{I} \;\middle|\; I_1 \subseteq I \text{ or } I_2 \subseteq I \right\} \Big)^{II} = \\
&= \Big(\left\{ I \in \mathbf{I} \;\middle|\; I_1 \cap I_2 \subseteq I \right\} \Big)^{II} \\
\mathbf{I}_3 &= \left\{ I \in \mathbf{I} \;\middle|\; I_3 \subseteq I \right\}
\end{aligned}$$

Since ε_I is a bijection, it holds that $(\mathbf{I}_1 \cup \mathbf{I}_2)^{II} = \mathbf{I}_3$ iff $I_1 \cap I_2 = I_3$.

Thus $\varepsilon_I : \underline{\mathfrak{B}} \longrightarrow \mathbf{V}_{Lb}$ is an isomorphism between $\langle \underline{\mathfrak{B}}, \vee, \wedge \rangle$ and $\langle \mathbf{V}_{Lb}, \cup, \cap \rangle$, i.e. $\langle \mathbf{V}_{Lb}, \cup, \cap \rangle$ is isomorphic to the free completely distributive lattice generated by $\langle \mathbf{C}_{Lb}, * \rangle$.

Let's check now that $\downarrow : \mathbf{C}_{Lb} \longrightarrow \mathbf{V}_{Lb}$ defined as is an injective homomorphism. Stumme has already showed that $\varepsilon_p : \mathbf{C}_{Lb} \longrightarrow \underline{\mathfrak{B}}$ given by

$$\varepsilon_p(G) \mapsto \Big(\left\{ F \in \mathbf{F} \;\middle|\; G \in F \right\}, \left\{ I \in \mathbf{I} \;\middle|\; G \in I \right\} \Big)$$

is an injective homomorphism between the partial lattice $\langle \mathbf{C}_{Lb}, +, * \rangle$ and $\langle \underline{\mathfrak{B}}, \vee, \wedge \rangle$. So that $\varepsilon_I \circ \varepsilon_p$ is an injective homomorphism between $\langle \mathbf{C}_{Lb}, +, * \rangle$ and $\langle \mathbf{V}_{Lb}, \cup, \cap \rangle$ given by

$$\varepsilon_I \circ \varepsilon_p(G) \mapsto \bigcap \{ \, I \in \mathbf{V}_{Lb} \mid G \in I \, \}$$

Note that for any causal graph G and $G' \in \mathbf{C}_{Lb}$ s.t. $G' \leq G$ it holds that $G' \in \varepsilon_I \circ \varepsilon_p(G)$ that is $\downarrow G \subseteq \varepsilon_I \circ \varepsilon_p(G)$. Furthermore for every causal graph G it holds that $\varepsilon(G)$ is an ideal, i.e. $\downarrow G \in \mathbf{V}_{Lb}$ and it is clear that $G \in \downarrow G$ so that, $\varepsilon_I \circ \varepsilon_p$ is a intersection of which one element is $\downarrow G$, thus $\varepsilon_I \circ \varepsilon_p(G) \subseteq \downarrow G$. That is $\downarrow G = \varepsilon_I \circ \varepsilon_p(G)$ and consequently it is an injective homomorphism between $\langle \mathbf{C}_{Lb}, +, * \rangle$ $\langle \mathbf{V}_{Lb}, \cup, \cap \rangle$.

Proof of Theorem 2. Since the set of causal values forms a lattice causal logic programs can be translated to *Generalized Annotated Logic Programming* (GAP). GAP is a general a framework for multivalued logic programming where the set of truth values must to form an upper semilattice and rules (*annotated clauses*) have the following form:

$$H : \rho \leftarrow B_1 : \mu_1 \ \& \ \ldots \ \& \ B_n : \mu_m \tag{3}$$

where $L_0, \ldots, L_m$ are literals, ρ is an *annotation* (may be just a truth value, an *annotation variable* or a *complex annotation*) and $\mu_1, \ldots, \mu_n$ are values or annotation variables. A complex annotation is the result to apply a total continuous function to a tuple of annotations. Thus a positive program P is encoded in a GAP program, GAP(P) rewriting each rule $R \in \Pi$ of the form

$$t : H \leftarrow B_1 \wedge \ldots \wedge B_n \tag{4}$$

as a rule GAP(R) in the form (3) where $\mu_1, \ldots, \mu_n$ are annotation that capture the causal values of each body literal and ρ is a complex annotation defined as $\rho = (\mu_1 * \ldots * \mu_n) \cdot t$.

Thus we will show that a causal interpretation $I \models \Pi$ if and only if $I \models^r$ GAP(P) where $\models^r$ refers to the GAP restricted semantics.

For any program P and interpretation I, by definition, $I \models P$ (resp. $I \models^r$ GAP(P)) iff $I \models R$ (resp. $I \models^r$ GAP(R)) for every rule $R \in P$. Thus it is enough to show that for every rule R it holds that $I \models R$ iff $I \models^r$ GAP(R).

By definition, for any rule R of the form of (4) and an interpretation I, $I \models R$ if and only if $\big(I(B_1) * \ldots * I(B_n)\big) \cdot t \leq I(H)$ whereas for any rule GAP(R) in the form of (3), $I \models^r GAP(R)$ iff for all $\mu_i \leq I(B_i)$ implies that $\rho = (\mu_1 * \ldots * \mu_n) \cdot t \leq I(H)$.

For the only if direction, take $\mu_i = I(B_i)$, then $\rho = (\mu_1 * \ldots * \mu_n) \cdot t = (I(B_1) * \ldots * I(B_n)) \cdot t$ and then $\rho \leq I(H)$ implies $\big(I(B_1) * \ldots * I(B_n)\big) \cdot t \leq I(H)$, i.e. $I \models^r GAP(R)$ implies $I \models R$. For the if direction, take $\mu_i \leq I(B_i)$ then, since product an applications are monotonic operations, it follows that $(\mu_1 * \ldots * \mu_n) \cdot t \leq (I(B_1) * \ldots * I(B_n)) \cdot t \leq I(H)$, That is, $I \models R$ also implies $I \models^r GAP(R)$. Consequently $I \models R$ iff $I \models^r GAP(R)$.

Thus, from Theorem 1 in (Kifer and Subrahmanian 1992), it follows that the operator T_P is monotonic.

To show that the operator T_P is also continuous we need to show that for every causal program P the translation GAP(P) is an *acceptable* program. Indeed since in a program GAP(P) all body atoms are v-annotated it is *acceptable*. Thus from Theorem 3 in (Kifer and Subrahmanian 1992), it follows that $T_P \uparrow^{\omega} (\mathbf{0}) = lfp(T_P)$ and this is the least model of P. Furthermore, since the lattice form by interpretations is finite, there is some positive interger n s.t. $T_P \uparrow^{n} (\mathbf{0}) = T_P \uparrow^{\omega} (\mathbf{0})$.

Lemma 7.1
Given a completely labelled program P, for every atom p and positive integer n it holds that

$$T_P{\uparrow}^{\,n}\,(\mathbf{0})(p) = \sum_{R\in\Psi}\sum_{f\in R}\prod\big\{\ f\big(T_P{\uparrow}^{\,k}\,(\mathbf{0})(q)\big)\ \big|\ q\in body(R)\ \big\}\cdot label(R)$$

where Ψ is the set of rules $\Psi = \{\ R\in\Pi\ \big|\ head(R)=p\ \}$ and R is the set of choice functions $R = \{\ f\ \big|\ f(S)\in S\ \}$.

Proof. By definition of $T_P{\uparrow}^{\,k}\,(\mathbf{0})(p)$ it follows that

$$T_P{\uparrow}^{\,n}\,(\mathbf{0})(p) = \sum\big\{\ \big(T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q_1)*\ldots*T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q_1)\big)\cdot label(R)\ \big|\ R\in P \text{ with } head(R)=p\ \big\}$$

then, applying distributive of application w.r.t. to the sum and and rewriting the sum and the product aggregating properly, it follows that

$$T_P{\uparrow}^{\,n}\,(\mathbf{0})(p) = \sum_{R\in\Psi}\prod\big\{\ T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q)\ \big|\ q\in body(R)\ \big\}\cdot label(R)$$

Furthermore for any atom q the causal value $T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q)$ can be expressed as the sum of all causes in it and then

$$T_P{\uparrow}^{\,n}\,(\mathbf{0})(p) = \sum_{R\in\Psi}\prod\big\{\ \sum_{f\in R} f\big(T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q)\big)\ \big|\ q\in body(R)\ \big\}\cdot abel(R)$$

and applying distributivity of products over sums it follows that

$$T_P{\uparrow}^{\,n}\,(\mathbf{0})(p) = \sum_{R\in\Psi}\sum_{f\in R}\prod\big\{\ f\big(T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q)\big)\ \big|\ q\in body(R)\ \big\}\cdot l_R \qquad \square$$

Lemma 7.2
Let P be a positive, completely labelled program and G be a cause. Then for every atom p it holds that $G\in T_P{\uparrow}^{\,n}\,(\mathbf{0})(p)$ iff there is a rule $l : p\leftarrow q_1,\ldots,q_m$ and causes G_{q_1}, ..., G_{q_m} respectively in $T_P{\uparrow}^{\,n}\,(\mathbf{0})(q_i)$ and $G\leq\big(G_{q_1}*\ldots*G_{q_m}\big)\cdot l$.

Proof. From Lemma 7.1 it follows that $G\in T_P{\uparrow}^{\,n}\,(\mathbf{0})(p)$ iff

$$G\in value\Big(\ \sum_{R\in\Psi}\sum_{f\in R}\prod\big\{\ f\big(T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q)\big)\ \big|\ q\in body(R)\ \big\}\cdot label(R)\ \Big)$$

iff

$$G\in\bigcup_{R\in\Psi}\bigcup_{f\in R} value\Big(\prod\big\{\ {\downarrow} f\big(T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q)\big)\Big)\ \big|\ q\in body(R)\ \big\}\cdot label(R)$$

iff there is $R\in\Phi$, with $head(R)=p$ and a choice function $f\in\Psi$ s.t.

$$G\in value\Big(\prod\big\{\ f\big(T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q)\big)\ \big|\ q\in body(R)\ \big\}\cdot label(R)\Big)$$

Let $R = l : p\leftarrow q_1,\ldots,q_m$ and $f\big(T_P{\uparrow}^{\,n-1}\,(\mathbf{0})(q_i)\big) = G_{q_i}$. Then the above can be rewritten as $G\leq\big(G_{q_1}*\ldots*G_{q_m}\big)\cdot l$. $\square$

Lemma 7.3
For any proof $\pi(p)$ it holds that

$$cause\Big(\frac{\pi(q_1),\dots,\pi(q_m)}{p}\,(l)\Big) = \big(cause(\pi(q_1)) * \dots * cause(\pi(q_1))\big) \cdot l$$

Proof. We proceed by structural induction assuming that for every proof in the antecedent $\pi(q_i)$ and every label $l' \in V(cause(\pi(q_i)))$ there is an edge $(l' label(\pi(q_i))) \in E(cause(\pi(q_i)))$.

By definition $cause(\pi(p)) = G^*_{\pi(p)}$ is the reflexive and transitive closure of $G_{\pi(p)}$ and then

$$cause(\pi(p)) = \Big(\bigcup\{\ cause(\pi(q_i)) \mid 1 \le i \le m\ \} \cup \{\ (label(\pi(q_i),l) \mid 1 \le i \le m\ \}\Big)^*$$

Thus, $cause(\pi(p)) \ge \prod\{\ cause(\pi(q_i)) \mid 1 \le i \le m\ \} \cdot l$ and remain to show that for every atom q_i and label $l' \in V(cause(\pi(q_i)))$ the edge $(l',l) \in E(cause(\pi(p)))$. Indeed, since by induction hypothesis there is an edge $(l', label(\pi(q_i))) \in E(cause(\pi(q_i))) \subseteq E(cause(\pi(p))$, the fact that the edge $(label(\pi(q_i),l) \in E(cause(\pi(p)))$ and since $cause(\pi(p))$ is closed transitively, it follows that $(l',l) \in E(cause(\pi(p)))$. □

Lemma 7.4
Let P be a positive, completely labelled program and $\pi(p)$ be a proof for p w.r.t. P. Then it holds that $cause(\alpha_p) \in T_P\!\uparrow^{h}(\mathbf{0})(p)$ where h is the height of $\pi(p)$ which is recursively defined as

$$height(\pi) = 1 + \max\{\ heigh(\pi') \mid \pi' \text{ is a sub-proof of } \pi\ \}$$

Proof. In case that $h = 1$ the antecedent of $\pi(p)$ is empty, i.e.

$$\pi(p) = \frac{\top}{p}\,(l)$$

where l is the label of the fact $(l : p)$. Then $casue(\pi(p)) = l$. Furthermore, since the fact $(l : p)$ is in the program P, it follows that $l \in T_P\!\uparrow^{1}(\mathbf{0})(p)$.

In the remain cases, we proceed by structural induction assuming that for every natural number $h \le n-1$, atom p and proof $\pi(p)$ of p w.r.t. P of which $height(\pi(p)) = h$ it holds that $casue(\pi(p)) \in T_P\!\uparrow^{h}(\mathbf{0})(p)$ and we will show it in case that $h = n$.

Since $heigh(\alpha_p) > 1$ it has a non empty antecedent, i.e.

$$\pi(p) = \frac{\pi(q_1),\dots,\pi(q_m)}{p}\,(l)$$

where l is the label of the rule $l : p \leftarrow q_1,\dots,q_m$. By *height* definition, for each q_i it holds that $heigh(\pi(q_i)) \le n-1$ and so that, by induction hypothesis, $casue(\pi(q_i)) \in T_P\!\uparrow^{h-1}(\mathbf{0})(q_i)$. Thus, from Lemmas 7.2 and 7.3, it follows respectively that

$$\prod\{\ casue(\pi(q_i)) \mid 1 \le i \le m\ \} \cdot l \in T_P\!\uparrow^{h}(\mathbf{0})(p)$$
$$cause(\pi(p)) = \prod\{\ casue(\pi(q_i)) \mid 1 \le i \le m\ \} \cdot l$$

That is, $cause(\pi(p)) \in T_P\!\uparrow^{h}(\mathbf{0})(p)$. □

Lemma 7.5
Let P be a positive, completely labelled program and G be a cause. Then for every atom p it holds that G is maximal in $T_P\!\uparrow^{n}(\mathbf{0})(p)$ iff there is a rule $l : p \leftarrow q_1, \dots, q_m$ and causes $G_{q_1}, \dots, G_{q_m}$ which are respectively maximal in $T_P\!\uparrow^{n}(\mathbf{0})(q_i)$ and $G = \big(G_{q_1} * \dots * G_{q_m}\big) \cdot l$

Proof. From Lemma 7.2 it follows that $G \in T_P\!\uparrow^{n}(\mathbf{0})(p)$ iff there is a rule $l : p \leftarrow q_1, \dots, q_m$ and causes $G_{q_1}, \dots, G_{q_m}$ respectively in $T_P\!\uparrow^{n-1}(\mathbf{0})(q_i)$ s.t. $G = \big(G_{q_1} * \dots * G_{q_m}\big) \cdot l$. Suppose that some G_{q_i} is not maximal, i.e. there is some maximal $G'_{q_i} \in T_P\!\uparrow^{n-1}(\mathbf{0})(q_i)$ s.t. $G_{q_i} < G'_{q_i}$. Then for the cause G' defined as $G' = (G_{q_1} * \dots * G_{q_{i-1}} * G'_{q_i} * G_{q_{i+1}} * \dots * G_{q_m}) \cdot l$ it holds that $G' \in T_P\!\uparrow^{n}(\mathbf{0})(p)$. Furthermore, by product and application monotonicity it holds that $G \leq G'$ and since G is maximal, it must to be that $G = G'$. That is, $G = \big(G_{q_1} * \dots * G_{q_m}\big) \cdot l$ where each G_{q_i} is maximal.

Lemma 7.6
Let P be a positive, completely labelled program and $\pi(p)$ be a proof of p w.r.t. P. For every atom p and maximal cause $G \in T_P\!\uparrow^{\omega}(\mathbf{0})(p)$ there is a non-redundant proof $\pi(p)$ for p w.r.t. P s.t. $\mathit{cause}(\pi(p)) = G$.

Proof. From Lemma 7.5 for any maximal cause $G \in T_P\!\uparrow^{n}(\mathbf{0})(p)$, there is a rule $l : p \leftarrow q_1, \dots, q_m$ and maximal causes $G_{q_1} \in T_P\!\uparrow^{h-1}(\mathbf{0})(q_1), \dots, G_{q_m} \in T_P\!\uparrow^{n-1}(\mathbf{0})(q_m)$ s.t.

$$G = (G_{q_1} * \dots * G_{q_m}) \cdot l$$

Furthermore, we assume as induction hypothesis that for every atom q_i there is a non redundant proof $\pi(q_i)$ for q_i w.r.t. P s.t. $\mathit{cause}(\pi(q_i)) = G_{q_i}$. Then $\pi(p)$ defined as

$$\pi(p) = \frac{\pi(q_1), \dots, \pi(q_m)}{p}\,(l)$$

is a proof for p w.r.t. P which holds $\mathit{cause}(\pi(p)) = G_p$ (from Lemma 7.3) and $\mathit{height}(\pi(p) \leq h$. Furthermore, suppose that $\pi(p)$ is redundant, i.e. there is a cause π' for p w.r.t P such that $\mathit{cause}(\pi(p)) < \mathit{cause}(\pi')$. Let n be a natural number grater that $\mathit{height}(\pi')$ and $\mathit{heigh}(\pi(p))$. Then, from Lemma 7.3, it follows that $\mathit{cause}(\pi') \in T_P\!\uparrow^{n}(\mathbf{0})(p)$, i.e. $\mathit{cause}(\pi') \in T_P\!\uparrow^{\omega}(\mathbf{0})(p)$ which contradicts the hypothesis that G is maximal in $T_P\!\uparrow^{\omega}(\mathbf{0})(p)$. □

Proof of Theorem 3. From Theorem 2 it follows that the least model I is equal to $T_P\!\uparrow^{\omega}(\mathbf{0})$. For the only if direction, from Lemma 7.6, it follows that for every maximal cause $G \in I(p) = T_P\!\uparrow^{\omega}(\mathbf{0})(p)$ there is a non-redundant proof $\pi(p)$ for p w.r.t P s.t. $G = \mathit{cause}(\pi(p))$. That is, $\pi(p) \in \Pi_p$ and then $G = \mathit{cause}(\pi(p)) \in \mathit{causes}(\Pi_p)$. For the if direction, from Lemma 7.4, for every $G \in \mathit{causes}(\Pi_p)$, i.e. $G = \mathit{causes}(\pi(p))$ for some non-redundant proof $\pi(p)$ for p w.r.t. P, it holds that $G \in T_P\!\uparrow^{\omega}(\mathbf{0})(p)$ and so that $G \in I(p)$. Furthermore, suppose that G is not maximal, i.e. there is a maximal cause $G' \in I(p)$ s.t. $G < G'$ and a proof π' for p w.r.t. P s.t. $\mathit{cause}(\pi') = G'$ which contradicts that $\pi(p)$ is non-redundant. □

Lemma 7.7
Let P, Q two positive causal logic programs such that Q is the result of replacing label l in P by some u (a label or 1) then $T_Q{\uparrow}^n(\mathbf{0})(p) = T_P{\uparrow}^n(\mathbf{0})(p)[l \mapsto u]$ for any positive integer n and atom p.

Proof. In case that $n = 0$ $T_Q{\uparrow}^n(\mathbf{0})(p) = 0$ and $T_P{\uparrow}^n(\mathbf{0})(p) = 0$ and $0 = 0[l \mapsto u]$. That is $T_Q{\uparrow}^n(\mathbf{0})(p) = T_P{\uparrow}^n(\mathbf{0})(p)[l \mapsto u]$.

We proceed by induction on n assuming that $T_Q{\uparrow}^{n-1}(\mathbf{0})(p) = T_P{\uparrow}^{n-1}(\mathbf{0})(p)[l \mapsto u]$ for any atom p and we will show that $T_Q{\uparrow}^n(\mathbf{0})(p) = T_P{\uparrow}^n(\mathbf{0})(p)[l \mapsto u]$.

Pick $G \in T_P{\uparrow}^n(\mathbf{0})(p)$ then, from Lemma 7.2, there is a rule $l' : p \leftarrow q_1, \dots, q_m$ and causes $G_{q_1}, \dots, G_{q_m}$ respectively in $T_P{\uparrow}^{n-1}(\mathbf{0})(q_i)$ s.t. $G \leq G_R = (G_{q_1} * \dots * G_{q_m}) \cdot l'$. Thus, by induction hypothesis, for every atom q_i and cause $G_{q_i} \in T_P{\uparrow}^{n-1}(\mathbf{0})(q)$ it holds that $G_{q_i}[l \mapsto u] \in T_Q{\uparrow}^{n-1}(\mathbf{0})(q_i)$. where $G_{q_i}[l \mapsto u]$ is just the result of replacing l by u when u is a label or removing l when $u = 1$.

Let $G_R[l \mapsto u]$ be a cause defined as $G_R[l \mapsto u] = \big(G_{q_1}[l \mapsto u] * \dots * G_{q_m}[l \mapsto u]\big) \cdot l'[l \mapsto u]$. Then, since $G \leq G_R$, it follows that $G[l \mapsto u] \leq G_R[l \mapsto u]$ and then, again from Lemma 7.2, it follows that $G[l \mapsto u] \in T_Q{\uparrow}^n(\mathbf{0})(p)$. That is $T_P{\uparrow}^n(\mathbf{0})(p)[l \mapsto u] \subseteq T_Q{\uparrow}^n(\mathbf{0})(p)$.

Pick $G \in T_Q{\uparrow}^n(\mathbf{0})(p)$ then, from Lemma 7.2, there is a rule there is a rule $l' : p \leftarrow q_1, \dots, q_m$ s.t. $G \leq G_R = (G_{q_1} * \dots * G_{q_m}) \cdot l'[l \mapsto u]$. By induction hypothesis, for every atom q_i and cause $G_{q_i} \in T_Q{\uparrow}^{n-1}(\mathbf{0})(q_i)$ it holds that $G_{q_i} \in T_P{\uparrow}^{n-1}(\mathbf{0})(q_i)[l \mapsto u]$, i.e. there is a cause $G'_{q_i} \in T_P{\uparrow}^{n-1}(\mathbf{0})(q_i)$ s.t. $G'_{q_i}[l \mapsto u] = G_{q_i}$. Let G'_R be a cause s.t. $G'_R = \big(G'_{q_1} * \dots * G'_{q_m}\big) \cdot l'$. From Lemma 7.2 for every cause $G' \leq G'_R$ it holds that $G' \in T_P{\uparrow}^n(\mathbf{0})(p)$. Since $G'_R[l \mapsto u] = G_R$ and $G \leq G_R$ it follows that $G \in T_P{\uparrow}^n(\mathbf{0})(p)[l \mapsto u]$. Consequently $T_P{\uparrow}^n(\mathbf{0})(p)[l \mapsto u] \supseteq T_Q{\uparrow}^n(\mathbf{0})(p)$ and then $T_P{\uparrow}^n(\mathbf{0})(p)[l \mapsto u] = T_Q{\uparrow}^n(\mathbf{0})(p)$.

Lemma 7.8
Let t be a causal term. Then $value(t[l \mapsto u]) = value(t)[l \mapsto u]$.

Proof. We proceed by structural induction. In case that t is a label. If $t = l$ then $value(l[l \mapsto u] = value(u) =\downarrow u = value(l)[l \mapsto u]$. If $t = l' \neq l$ then $value(l'[l \mapsto u] = value(l') =\downarrow l' = value(l')[l \mapsto u]$. In case that $t = \prod T$ it follows that $value(\prod T[l \mapsto u]) = \bigcap\{\ value(t'[l \mapsto u]) \mid t' \in T\ \})$ and by induction hypothesis $value(t'[l \mapsto u]) = value(t')[l \mapsto u]$. Then $value(\prod T[l \mapsto u]) = \bigcap\{\ value(t')[l \mapsto u] \mid t' \in T\ \}) = value(\prod T)[l \mapsto u]$. The cases for $t = \sum T$ is analogous. In case that $t = t_1 \cdot t_2$ it follows that $value(t[l \mapsto u]) = value(t_1[l \mapsto u]) \cdot value(t_2[l \mapsto u]) = value(t_1)[l \mapsto u] \cdot value(t_2)[l \mapsto u] = value(t)[l \mapsto u]$

Proof of Theorem 4. From Theorem 2 there is some positive integer n s.t. $T_P{\uparrow}^n(\mathbf{0})$ and $T_{P'}{\uparrow}^n(\mathbf{0})$ are equal to the least model of P and P' respectively. Furthermore, from Lemma 7.7, it follows that $T_P{\uparrow}^n(\mathbf{0})(p) = T_{P'}{\uparrow}^n(\mathbf{0})(p)$ for any atom p. Lemma 7.8 shows that the replacing can be done in any causal term of which valuation is equivalent to them.

Proof of Theorem 5. It is clear that if every rule in P is unlabelled, i.e. $P = P'$, then their least model assigns 0 to every *false* atom and 1 to every *true* atom, so that their least models coincide with the classical one, i.e. $I = I'$ and then $I^{cl} = I = I'$. Otherwise, let P_n be a program where n

rules are labelled. We can build a program P_{n-1} removing one label l and, from Theorem 4, it follows that $I_{n-1} = I_n[l \rightarrow 1]$. By induction hypothesis the corresponding classical interpretation of least model of P_{n-1} coincides with the least model of the unlabelled program, i.e. $I^{cl}_{n-1} = I'$, and then $I_n[l \mapsto 1]^{cl} = I^{cl}_{n-1} = I'$. Furthermore, for every atom p and cause G it holds that $G \in I_n(p)$ iff $G[l \mapsto 1] \in I_n[l \mapsto 1](p)$. Simple remain to note that $value(0) = \emptyset$, so that $I_n(p) = 0$ iff $I_n[l \mapsto 1](p) = 0$ and consequently $I^{cl}_n = I_n[l \mapsto 1]^{cl} = I'$. $\square$

Proof of Theorem 6. By definition I and I^{cl} assigns 0 to the same atoms, so that $P^I = P^{I^{cl}}$. Furthermore let Q (instead of P' for clarity) be the unlabelled version of P. Then $Q^{I^{cl}}$ is the unlabelled version of P^I. (1) Let I be a stable model of P and J be the least model of $Q^{I^{cl}}$. Then, I is the least model of P^I and, from Theorem 5, it follows that $I^{cl} = J$, i.e. I^{cl} is a stable model of Q. (2) Let I' is a stable model of Q and I be the least model of $P^{I'}$. Since I' is a stable model of Q, by definition it is the least model of $Q^{I'}$, furthermore, since $Q^{I'}$ is the unlabelled version of $P^{I'}$ it follows, from Theorem 5, that $I^{cl} = I'$. Note that $P^I = P^{I^{cl}} = P^{I'}$. Thus I is a stable model of P. $\square$