Enablers and Inhibitors in Causal Justifications of Logic Programs

Pedro Cabalar and Jorge Fandinno

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

Abstract. In this paper we propose an extension of logic programming (LP) where each default literal derived from the well-founded model is associated a justification represented as an algebraic expression. This expression contains both causal explanations (in the form of proof graphs built with rule labels) and terms under the scope of negation that stand for conditions that enable or disable the application of causal rules. Using some examples, we discuss how these new conditions, we respectively call *enablers* and *inhibitors*, are intimately related to default negation and have an essentially different nature from regular cause-effect relations. The most important result is a formal comparison to the recent algebraic approaches for justifications in LP: *Why-not Provenance* (WnP) and *Causal Graphs* (CG). We show that the current approach extends both WnP and CG justifications under the Well-Founded Semantics and, as a byproduct, we also establish a formal relation between these two approaches.

1 Introduction

The strong connection between Non-Monotonic Reasoning (NMR) and Logic Programming (LP) semantics for default negation has made possible that LP tools became nowadays an important paradigm for Knowledge Representation (KR) and problem-solving in Artificial Intelligence (AI). In particular, *Answer Set Programming* (ASP) [1, 2] has raised as a preeminent LP paradigm for practical NMR with applications in diverse areas of AI including planning, reasoning about actions, diagnosis, abduction and beyond. The ASP semantics is based on *stable models* [3] and is also closely related to the other mainly accepted interpretation for default negation, *well-founded* semantics (WFS) [4]. One interesting difference between these two LP semantics and classical models (or even other NMR approaches) is that true atoms in LP must be founded or justified by a given derivation. These *justifications* are not provided in the semantics itself, but can be syntactically built in some way in terms of the program rules, as studied in several approaches [5–11].

Rather than manipulating justifications as syntactic objects, two recent approaches have considered multi-valued extensions of LP where justifications are treated as *algebraic* constructions: *Why-not Provenance* (WnP) [12] and *Causal Graphs* (CG) [13]. Although these two approaches present formal similarities, they start from different understandings of the idea of justification. On the one hand, WnP answers the query of

why some literal L might hold by providing conjunctions of "hypothetical modifications" on the program that would allow deriving L. These modifications include rule labels, expressions like $not(A)$ with A an atom, or negations '$\neg$' of the two previous cases. As an example, a justification for L like $r_1 \wedge not(p) \wedge \neg r_2 \wedge \neg not(q)$ means that the presence of rule r_1 and the absence of atom p would allow deriving L if both rule r_2 were removed and atom q were added to the program. If we want to explain why L *actually* holds, we have to restrict to justifications without '$\neg$', that is, those without program modifications (which will be the focus of this paper).

On the other hand, CG-justifications start from identifying program rules as *causal laws* so that, for instance, $(p \leftarrow q)$ can be read as "event q *causes* effect p." Under this viewpoint, (positive) rules offer a natural way for capturing the concept of *causal production*, i.e. a continuous chain of events that has helped to cause or produce an effect [14, 15]. The explanation of a true atom is made in terms of graphs formed by rule labels that reflect the ordered rule applications required for deriving that atom. These graphs are obtained by algebraic operations exclusively applied on the positive part of the program. Default negation in CG is understood as absence of cause and, consequently, a false atom has no justification.

The explanation of an atom A in CG is more detailed than in WnP, since the former contains graphs that correspond to all relevant proofs of A whereas in WnP we just get conjunctions that do not reflect any particular ordering among rule applications. However, as explained before, CG does not reflect the effect of default negation in a given derivation and, sometimes, this information is very valuable, especially if we want to answer questions of the form "why not."

To understand the kind of problems we are interested in, consider the following example. A drug d in James Bond's drink causes his paralysis p provided that he was not given an antidote a that day. We know that Bond's enemy, Dr. No, poured the drug:

$$p \leftarrow d, not\ a \tag{1}$$
$$d \tag{2}$$

In this case it is obvious that d causes p, whereas the absence of a just *enables* the application of the rule. Now, suppose we are said that Bond is daily administered an antidote by the MI6, unless it is a holiday h:

$$a \leftarrow not\ h \tag{3}$$

Adding this rule makes a to become an *inhibitor* that prevents d to cause p. But suppose now that we are in a holiday, that is, fact h is added to the program (1)-(3). Then, the inhibitor a is *disabled* and d causes p again. However, we do not consider that the holiday h is a (productive) cause for Bond's paralysis p although, indeed, the latter counterfactually depends on the former: "had not been a holiday h, Bond would have not been paralysed." We will say that the fact h, which disables an inhibitor of d, is an *enabler* of d.

In this work we propose dealing with these concepts of enablers and inhibitors by augmenting CG justifications with a new negation operator '$\sim$' in the CG causal algebra. We show that this new approach, we call *Extended Causal Justifications* (ECJ)

captures WnP justifications under the Well-founded Semantics and, as a byproduct, we establish a formal relation between WnP and CG.

The rest of the paper is structured as follows. The next section defines the new approach. Sections 3 and 4 explain through a running example the formal relations to CG and WnP, respectively. The next section discusses some related work and, finally, Section 6 concludes the paper.

2 Extended Causal Justifications (ECJ)

A *signature* is a pair $\langle At, Lb\rangle$ of sets that respectively represent *atoms* (or *propositions*) and *labels*. Intuitively, each atom in At will be assigned justifications built with rule labels from Lb. These justifications will be expressions that combine four different algebraic operators: a product '$*$' representing conjunction or joint causation; a sum '$+$' representing alternative causes; a non-commutative product '$\cdot$' that captures the sequential order that follows from rule applications; and a non-classical negation '$\sim$' which will precede inhibitors (negated labels) and enablers (doubly negated labels).

Definition 1 (Term). *Given a set of labels Lb, a* term*, t is recursively defined as one of the following expressions $t ::= l \mid \prod S \mid \sum S \mid t_1 \cdot t_2 \mid \sim t_1$ where $l \in Lb$, t_1, t_2 are in their turn terms and S is a (possibly empty and possibly infinite) set of terms. A term is* elementary *if it has the form $\sim\sim l$, $\sim l$ or l with $l \in Lb$ being a label.* □

When $S = \{t_1, \ldots, t_n\}$ is finite we simply write $\prod S$ as $t_1 * \cdots * t_n$ and $\sum S$ as $t_1 + \cdots + t_n$. Moreover, when $S = \emptyset$, we denote $\prod S$ by 1 and $\sum S$ by 0, as usual, and these will be the identities of the product '$*$' and the addition '$+$', respectively. We assume that '$\cdot$' has higher priority than '$*$' and, in its turn, '$*$' has higher priority than '$+$'.

pseudo-complement	*De Morgan*	*excluded middle*	*appl. negation*
$t * {\sim}t = 0$	${\sim}(t+u) = ({\sim}t * {\sim}u)$	${\sim}t + {\sim}{\sim}t = 1$	${\sim}(t \cdot u) = {\sim}(t * u)$
${\sim}{\sim}{\sim}t = {\sim}t$	${\sim}(t * u) = ({\sim}t + {\sim}u)$		

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

Indempotence	*Addition distributivity*	*Product distributivity*
$l \cdot l = l$	$t \cdot (u+w) = (t \cdot u) + (t \cdot w)$	$c \cdot d \cdot e = (c \cdot d) * (d \cdot e)$ 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. 1. Properties of the '$\sim$' and '$\cdot$'operators (c, d, e are terms without '$+$' and l is a label).

Definition 2 (Value). *A* (causal) value *is each equivalence class of terms under axioms for a completely distributive (complete) lattice with meet '$*$' and join '$+$' plus the axioms of Figure 1. The set of (causal) values is denoted by $\mathbf{V}_{Lb}$.* □

Note that $\langle \mathbf{V}_{Lb}, +, *, \sim, 0, 1\rangle$ forms a pseudo-complemented, completely distributive (complete) lattice whose meet and join are, as usual, the product '$*$' and the addition '$+$'. Note also that all three operations, '$*$', '$+$' and '$\cdot$' are associative. Product '$*$'

and addition '$+$' are also commutative, and they hold the usual absorption and distributive laws with respect to infinite sums and products of a completely distributive lattice. We say that a term is in *negation normal form* (NNF) if no operators are in the scope of negation '$\sim$'. Without loss of generality, we assume from now on that all terms are in NNF. The lattice order relation is defined as usual in the following way:

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

Consequently 1 and 0 are respectively the top and bottom elements with respect to the $\leq$ order relation.

Definition 3 (Labelled logic program). *Given a signature $\langle At, Lb \rangle$, a* (labelled logic) program *P is a set of rules of the form:*

$$r_i : \ H \leftarrow B_1, \ldots, B_m, \text{ not } C_1, \ldots, \text{ not } C_n \tag{4}$$

where $r_i \in Lb$ is a label or $r_i = 1$, H (the head *of the rule) is an atom and B_i's and C_i's (the* body *of the rule) are either atoms or terms.* □

When $n = 0$ we say that the rule is positive, furthermore, if $m = 0$ we say that the rule is a *fact* and omit the symbol '$\leftarrow$.' When $r_i \in Lb$ we say that the rule is labelled; otherwise $r_i = 1$ and we omit both r_i and ':'. By these conventions, for instance, an unlabelled fact A is actually an abbreviation of $(1 : A \leftarrow)$. A program P is *positive* when all its rules are positive, i.e. it contains no default negation. It is *uniquely labelled* when each rule has a different label or no label at all. In this paper, we will we assume that programs are uniquely labelled. Furthermore, for clarity sake, we also assume that, for every atom $A \in At$, there is an homonymous label $A \in Lb$, and that each fact A in the program actually stands for the labelled rule $(A : \ A \leftarrow)$. For instance, following these conventions, a possible labelled version for the James Bond's program could be program P_1 below:

$$\begin{array}{lll} r_1 : & p \leftarrow d, not\ a & d \\ r_2 : & a \leftarrow not\ h & h \end{array}$$

where facts d and h stand for rules $(d : \ d \leftarrow)$ and $(h : \ h \leftarrow)$, respectively.

A *CP-interpretation* is a mapping $I : At \longrightarrow \mathbf{V}_{Lb}$ assigning a value to each atom. For interpretations I and J we say that $I \leq J$ when $I(A) \leq J(A)$ for each 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 each atom A to 0 (resp. 1). The value assigned to a negative literal *not* A by an interpretation I, denoted as $I(not\ A)$, is defined as: $I(not\ A) \stackrel{\text{def}}{=} \sim I(A)$. Similarly, for a term t, $I(t) \stackrel{\text{def}}{=} [t]$ is the equivalence class of t.

Definition 4 (Reduct). *Given a program P and an interpretation I we denote by P^I the positive program containing a rule like*

$$r_i : H \leftarrow B_1, \ldots, B_m, \ I(\text{not } C_1), \ldots, \ I(\text{not } C_n) \tag{5}$$

per each rule of the form (4) *in P.*

Definition 5 (Model). *An interpretation J is a (causal)* model *of a rule like* (5) *iff*

$$\big(J(B_1) * \ldots * J(B_m) * I(\text{not } C_1) * \ldots * I(\text{not } C_n) \big) \cdot r_i \ \leq \ J(H)$$

and it is a model of P^I, written $J \models P^I$, iff it is a model of all rules in P^I. The operator $\Gamma_P(I)$ returns the least model of the positive program P^I.

Program P^I is positive and, as happens in standard logic programming, it also has a *least* causal model. Furthermore the operator Γ_P is anti-monotonic, and therefore Γ_P^2 is monotonic having a least fixpoint $\mathbb{L}_P$ and a greatest fixpoint $\mathbb{U}_P \stackrel{\text{def}}{=} \Gamma_P(\mathbb{L}_P)$ that respectively correspond to the justifications for true and for non-false atoms in the (standard) well-founded model (WFM), we denote W_P. A *query literal (q-literal) L* is either an atom A, its default negation '*not* A' or the expression '*undef* A' meaning that A is undefined.

Definition 6 (Causal well-founded model). *Given a program P, the* causal well-founded model $\mathbb{W}_P$ *is a mapping from q-literals to values s.t.*

$$\mathbb{W}_P(A) \stackrel{\text{def}}{=} \mathbb{L}_P(A) \quad \mathbb{W}_P(\text{not } A) \stackrel{\text{def}}{=} \sim\mathbb{U}_P(A) \quad \mathbb{W}_P(\text{undef } A) \stackrel{\text{def}}{=} \sim\mathbb{W}_P(A) * \sim\mathbb{W}_P(\text{not } A) \ \square$$

As we will formalise below, when A is undefined in the standard well-founded model, $\mathbb{L}_P(A) \neq \mathbb{U}_P(A)$ and, thus, $\mathbb{W}_P(A) \neq 0$. Continuing with our running example, the causal WFM of program P_1 corresponds to $\mathbb{W}_{P_1}(d) = d$, $\mathbb{W}_{P_1}(h) = h$, $\mathbb{W}_{P_1}(a) = \sim h \cdot r_2$ and $\mathbb{W}_{P_1}(p) = (\sim\sim h * d) \cdot r_1 + (\sim r_2 * d) \cdot r_1$. Intuitively $(\sim\sim h * d) \cdot r_1$ means that the fact h (double negated label $\sim\sim h$) has enabled d (non negated label) to produce p by means of rule r_1. In its turn, $(\sim r_2 * d) \cdot r_1$ means that $d \cdot r_1$ would have been sufficient, had not been present r_2. Furthermore, $\mathbb{W}_{P_1}(a)$ means that a does not hold because the fact h (negated label $\sim h$) has inhibited rule r_2 to produce it. The following definitions formalise these concepts.

Let l be a label occurrence in a term t in the scope of $n \geq 0$ negation $\sim$ operators. We say that l is a *odd* or an *even* occurrence if n is odd or even, respectively. We further say that it is *strictly even* if it is even and $n > 0$.

Definition 7 (Justification). *Given a program P and a q-literal L we say that a term with no sums E is a* (sufficient causal) justification *for L iff $E \leq \mathbb{W}_P(L)$. Odd (resp. strictly even) labels*[1] *in E are called* inhibitors *(resp.* enablers*) of E. A justification is said to be* inhibited *if it contains some inhibitor and it is said to be* enabled *otherwise.* $\square$

For instance, in our previous example, there are two justifications, $E_1 = (\sim\sim h * d) \cdot r_1$ and $E_2 = (\sim r_2 * d) \cdot r_1$, for atom p. Justification E_1 is enabled because it contains no inhibitors (in fact, E_1 is the unique real support for p). Moreover, h is an enabler in E_1 because it is strictly even (it is in the scope of double negation). On the contrary, E_2 is disabled because it contains the inhibitor r_2 (because it occurs here in the scope of one negation). Intuitively, r_2 has prevented $d \cdot r_1$ to become a justification of p. The next theorem shows that the literals satisfied by the standard WFM are precisely those ones containing at least one enabled justification in the causal WFM.

[1] We just mention labels, and not their occurrences because terms are in NNF and E contains no sums: having odd and even occurrences of a same label would mean that $E = 0$.

Theorem 1. *Let P be a program and W_P its (standard) well-founded model. A q-literal L holds with respect to W_P if and only if there is some enabled justification E of L, that is $E \leq \mathbb{W}_P(L)$ and E does not contain (odd) negative labels.* □

Back to our example program P_1, as we had seen, atom p had an enabled justification $(\sim\sim h * d)\cdot r_1$. The same happens for atoms d and h whose respective justifications are just their own atom labels. Therefore, these three atoms hold in the standard WFM, W_{P_1}. On the contrary, as we discussed before, the only justification for a is inhibited by h, and thus, a does not hold in W_{P_1}. We can further check that a is false in W_{P_1} (it is not undefined) because literal *not a* holds, since $\mathbb{W}_{P_1}(\mathit{not}\ a) = \sim\sim h + \sim r_2$ provides two justifications, being the first one, $\sim\sim h$, enabled (it contains no inhibitors). The interest of an inhibited justification for a literal is to point out "potential" causes that have been prevented by some abnormal situation. In our case, the presence of $\sim h$ in $\mathbb{W}_{P_1}(a) = \sim h\cdot r_2$ points out that an exception h has prevented r_2 to cause a. When the exception is removed, the inhibited justification (after removing the inhibitors) becomes an enabled justification r_2 for a.

Theorem 2. *Let E be an inhibited justification of some atom A with respect to program P. Let Q be the result of removing from P all rules r_i whose labels are inhibitors in E. Similarly, let F be the result of removing those inhibitors $\sim r_i$ from E. Then F is an enabled justification of A with respect to Q.* □

3 Relation to Causal Graph Justifications

We discuss now the relation between ECJ and CG approaches. Formally, ECJ extends CG causal terms by the introduction of the new negation operator '$\sim$'. Semantically, however, there are more differences than a simple syntactic extension. A first minor difference is that ECJ is defined in terms of a WFM, whereas CG defines (possibly) several causal stable models. In the case of stratified programs, this difference is irrelevant, since the WFM is complete and coincides with the unique stable model. A second, more important difference is that CG exclusively considers productive causes in the justifications, disregarding additional information like the inhibitors or enablers from ECJ. As a result, a false atom in CG has *no justification* – its causal value is 0 because there was no way to derive the atom. For instance, in program P_1, the only CG stable model I just makes $I(a) = 0$ and we lose the inhibited justification $\sim h \cdot r_2$ (default r_2 could not be applied). True atoms like p also lose any information about enablers: $I(p) = d\cdot r_1$ and nothing is said about $\sim\sim h$. Another consequence of the CG orientation is that negative literals *not A* are never assigned a cause (different from 0 or 1), since they cannot be "derived" or produced by rules. In the example, we simply get $I(\mathit{not}\ a) = 1$ and $I(\mathit{not}\ p) = 0$.

To further illustrate the similitudes and differences between ECJ and CG, consider the following program P_2 capturing a variation of the Yale Shooting Scenario.

$d_{t+1} : dead_{t+1}$	$\leftarrow shoot_t,\ loaded_t,\ not\ ab_t$	$\overline{loaded}_0$	$load_1$
$l_{t+1} : loaded_{t+1}$	$\leftarrow load_t$	$\overline{dead}_0$	$water_3$
$a_{t+1} : ab_{t+1}$	$\leftarrow water_t$	$\overline{ab}_0$	$shoot_8$

plus the following rules corresponding inertia axioms

$$F_{t+1} \leftarrow F_t,\ not\ \overline{F}_{t+1} \qquad \overline{F}_{t+1} \leftarrow \overline{F}_t,\ not\ F_{t+1}$$

for $F \in \{loaded, ab, dead\}$. Atoms of the form $\overline{A}$ represent the strong negation of A and we assume we disregard models satisfying both A and $\overline{A}$. Atom $dead_9$ does not hold in the standard WFM of P_2, and so there is no CG-justification for it. Note here the importance of default reasoning. On the one hand, the default flow of events is that the turkey, Fred, continues to be alive when nothing threats him. Hence, we do not need a cause to explain why Fred is alive. On the other hand, shooting a loaded gun would normally kill Fred, being this a cause of its death. But, in this example, another exceptional situation – *water* spilled out – has *inhibited* this existing threat and allowed the world to flow as if nothing had happened (that is, following its default behaviour).

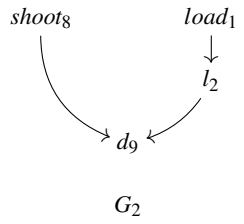
In the CG-approach, $dead_9$ is simply false by default and no justification is provided. However, a gun shooter could be "disappointed" since another conflicting default (shooting a loaded gun *normally* kills) has not worked. Thus, an expected answer for the shooter's question "why *not* $dead_9$?" is that $water_3$ broke the default, disabling d_9. In fact, ECJ yields the following inhibited justification for $dead_9$:

$$\mathbb{W}_{P_2}(dead_9) = (\sim water_3 * shoot_8 * load_1 \cdot l_2) \cdot d_9 \tag{6}$$

meaning that $dead_9$ could not be derived because of inhibitor $water_3$ prevented the application of r_1 to cause the death of Fred. Moreover, according to Theorem 2, if we remove fact $water_3$ (the inhibitor) from P_2, then for the new program P_3 we get:

$$\mathbb{W}_{P_3}(dead_9) = (shoot_8 * load_1 \cdot l_2) \cdot d_9 \tag{7}$$

which is nothing else but the result of removing $\sim water_3$ from (6). In fact, the only CG stable model of P_3 makes this same assignment (7) which also corresponds to the causal graph depicted in Figure 2. In the general case: CG-justifications intuitively correspond to enabled justifications after forgetting all the enablers. We formalise next the correspondence between CG and ECJ justifications.



G_2

Fig. 2. Cause of $dead_9$ in program P_2.

Definition 8 (Causal values). *A* CG term*, t is a term with no negation '*$\sim$*'.* CG values *are the equivalence classes of CG terms under the axioms of Definition 2. The set of*

CG values is denoted by $\mathbf{C}_{Lb}$*. We also define a mapping* $\lambda^c : \mathbf{V}_{Lb} \longrightarrow \mathbf{C}_{Lb}$ *from values into CG values in the following recursive way:*

$$\lambda^c(t) \stackrel{\text{def}}{=} \begin{cases} \lambda^c(u) \odot \lambda^c(w) & \text{if } t = u \odot v \text{ with } \odot \in \{+, *, \cdot\} \\ 1 & \text{if } t = \sim\sim l \text{ with } l \in Lb \\ 0 & \text{if } t = \ \sim l \text{ with } l \in Lb \\ l & \text{if } t = \ \ \ l \text{ with } l \in Lb \end{cases}$$

Note that we have assumed that t *is in negation normal form. Otherwise* $\lambda^c(t) \stackrel{\text{def}}{=} \lambda^c(u)$ *where* u *is the equivalent term in negation normal form.* □

Function λ^c maps every negated label $\sim l$ to 0 (which is the annihilator of both product '$*$' and application '$\cdot$' and the identity of addition '$+$'). Hence λ^c removes all the inhibited justifications. Furthermore λ^c maps every doubly negated label $\sim\sim l$ to 1 (which is the identity of both product '$*$' and application '$\cdot$'). Therefore λ^c removes all the enablers (i.e doubly negated labels $\sim\sim l$) for the remaining (i.e. enabled) justifications.

Definition 9 (CG stable models). *Given a program P, a* CG stable model *is a mapping* $\tilde{I} : At \longrightarrow \mathbf{C}_{Lb}$ *from atoms to CG values such that there exists a fixpoint I of the operator* Γ_P^2 *satisfying* $\lambda^c(I(A)) = \lambda^c(\Gamma_P(I(A)))$ *and* $\tilde{I}(A) \stackrel{\text{def}}{=} \lambda^c(I(A))$ *for every atom A.* □

Theorem 3. *For any program P, the CG values (Definition 8) and the CG stable models (Definition 9) are exactly the causal values and causal stable models defined in [13].* □

Theorem 3 shows that Definition 9 is an alternative definition of CG causal stable models. Furthermore, it settles that every causal model corresponds to some fixpoint of the operator Γ_P^2. Therefore, for every enabled justification there is a corresponding CG-justification common to all stable models. In order to formalise this idea we just take the definition of causal explanation from [16]. A graph of labels is a *causal graph* (c-graph) if it is a directed graph, transitively and reflexively closed. Furthermore we also define a one-to-one correspondence between c-graphs and causal values.

$$value(G) \stackrel{\text{def}}{=} \prod \{\ v_1 \cdot v_2 \ \mid\ (v_1, v_2) \text{ is an edge of } G\ \}$$

Definition 10 (CG-justification). *Given an interpretation I we say that a c-graph G is a* (sufficient) CG-justification *for an atom A iff* $value(G) \leq \tilde{I}(A)$. □

Note that mapping $value(\cdot)$ is a one-to-one correspondence and, thus, we can define $graph(v) \stackrel{\text{def}}{=} value^{-1}(v)$ for all $v \in \mathbf{C}_{Lb}$.

Theorem 4. *Let P be a program. For any enabled justification E of some atom A w.r.t.* $\mathbb{W}_P$*, i.e.* $E \leq \mathbb{W}_P(A)$*, there is a CG-justification* $G \stackrel{\text{def}}{=} graph(\lambda^c(E))$ *of A with respect to any stable model* $\tilde{I}$ *of P.* □

As happens between the (standard) Well-founded and Stable Model semantics, the converse of Theorem 4 does not hold in general. For instance, let P_4 be the program consisting on the following rules:

$$r_1 : a \leftarrow not\ b \qquad r_2 : b \leftarrow not\ a, not\ c \qquad c \qquad r_3 : c \leftarrow a \qquad r_4 : d \leftarrow b, not\ d$$

The (standard) WFM of P_4 is two-valued and corresponds to the unique (standard) stable model $\{a,c\}$. Furthermore there are two causal explanations of c with respect to this unique stable model: the fact c and the pair of rules $r_1 \cdot r_3$. Note that when c is removed $\{a,c\}$ is still the unique stable model, but all atoms are undefined in the WFM. Hence, $r_1 \cdot r_3$ is a justification with respect to the unique stable model of the program, but not with respect to is WFM.

4 Relation to Why-not Provenance

An evident similarity between ECJ and WnP is the use of an alternating fixpoint operator [17] which has been actually borrowed from WnP. However there are some slight differences. A first one is that we have incorporated from CG the non-commutative operator '$\cdot$' which allows capturing, not only which rules justify a given atom, but also the dependencies among these rules. The second is the use of a *non-classical* negation '$\sim$' that is crucial to distinguish between productive causes and enablers, something that cannot be represented with the classical negation '$\neg$' in WnP since double negation can always be removed. Apart from the interpretation of negation in both formalisms, there are other differences too. As an example, let us compare the justifications we obtain for $dead_9$ in program P_3. While for ECJ we obtained (7) (or graph G_2 in Figure 2), the corresponding WnP justification has the form:

$$l_2 \wedge d_9 \wedge load_1 \wedge shoot_8 \wedge not(water_0) \wedge not(water_1) \wedge \ldots \wedge not(water_7) \qquad (8)$$

A first observation is that the subexpression $l_2 \wedge d_9 \wedge load_1 \wedge shoot_8$ constitutes, informally speaking, a "flattening" of (7) (or graph G_2) where the ordering among rules has been lost. We get, however, new labels in the form of $not(A)$ meaning that atom A is required not to be a program fact, something that is not present in CG-justifications. For instance, (8) points out that *water* can not be spilt on the gun along situations $0,\ldots,7$. Although this information can be useful for debugging (the original purpose of WnP) its inclusion in a causal explanation is obviously inconvenient from a Knowledge Representation perspective, since it explicitly *enumerates all the defaults* that were applied (no water was spilt at any situation) something that may easily blow up the (causally) irrelevant information in a justification.

An analogous effect happens with the enumeration of exceptions to defaults, like inertia. Take program P_5 obtained from P_2 by removing all the performed actions, i.e., facts $load_1$, $water_3$, and $shoot_7$. As expected, Fred will be alive, $\overline{dead}_t$, at any situation t by inertia. ECJ will assign no cause for $dead_t$, not even any inhibited one, i.e. $\mathbb{W}_P(\overline{dead}_t) = 1$ and $\mathbb{W}_P(dead_t) = 0$ for any t. However, there are many WnP justifications of $dead_t$ corresponding to *all the plans* for killing Fred in t steps. For instance, among others, all the following:

$$d_9 \wedge \neg not(load_0) \wedge r_2 \wedge \neg not(shoot_1) \wedge not(water_0)$$
$$d_9 \wedge \neg not(load_0) \wedge r_2 \wedge \neg not(shoot_2) \wedge not(water_0) \wedge not(water_1)$$
$$d_9 \wedge \neg not(load_1) \wedge r_2 \wedge \neg not(shoot_3) \wedge not(water_0) \wedge not(water_1) \wedge not(water_2)$$
$$\ldots$$

are WnP-justifications for $dead_9$. The intuitive meaning of expressions of the form $\neg not(A)$ is that $dead_9$ can be justified by adding A as a fact to the program. For instance, the first conjunction means that it is possible to justify $dead_9$ by adding the facts $load_0$ and $shoot_1$ and not adding the fact $water_0$. We will call these justifications, which contain a subterm of the form $\neg not(A)$, *hypothetical* in the sense that they involve some hypothetical program modification.

Definition 11 (Provenance values). *A* provenance term*, t is recursively defined as one of the following expressions $t ::= l \mid \prod S \mid \sum S \mid \neg t_1$ where $l \in Lb$, t_1, t_2 are in their turn provenance terms and S is a (possibly empty and possible infinite) set of provenance terms.* Provenance values *are the equivalence classes of provenance terms under the equivalences of the Boolean algebra. We denote by $\mathbf{B}_{Lb}$ the set of provenance values over Lb.* □

Informally speaking, with respect to ECJ, we have removed the application '$\cdot$' operator, whereas product '$*$' and addition '$+$' hold the same equivalences as in Definition 2 and negation '$\sim$' has been replaced by '$\neg$' from Boolean algebra. Thus, '$\neg$' is classical and satisfies all the axioms of '$\sim$' plus $\neg\neg t = t$. Note also that, we have followed the convention from [12] of using the symbols '$\wedge$' instead of '$*$' to represent the meet and '$\vee$' instead of '$+$' to represent the join when we write provenance formulae. We define a mapping $\lambda^p : \mathbf{V}_{Lb} \longrightarrow \mathbf{B}_{Lb}$ in the following recursive way:

$$\lambda^p(t) \stackrel{\text{def}}{=} \begin{cases} \lambda^p(u) \odot \lambda^p(w) & \text{if } t = u \odot v \text{ with } \odot \in \{+, *\} \\ \lambda^p(u) * \lambda^p(w) & \text{if } t = u \cdot v \\ \neg\lambda^p(u) & \text{if } t = \sim u \\ l & \text{if } t = l \text{ with } l \in Lb \end{cases}$$

Definition 12 (Provenance). *Given a program P the why-not provenance program $\mathfrak{P}(P) \stackrel{\text{def}}{=} P \cup P'$ where P' contains a labelled fact of the form of $(\sim not(A) : A)$ for each atom $A \in At$ not occurring in P as a fact. We will write $\mathfrak{P}$ instead of $\mathfrak{P}(P)$ when the program P is clear by the context. We denote by $Why_P(L) \stackrel{\text{def}}{=} \lambda^p(\mathbb{W}_{\mathfrak{P}}(L))$ the why-not provenance of a q-literal L.* □

Theorem 5. *For any program P, the provenance of a literal according to Definition 12 is equivalent to the provenance defined in [12].* □

Theorem 5 shows that the provenance of a literal can obtained from replacing the negation '$\sim$' by '$\neg$' and '$\cdot$' by '$*$' in the causal WFM of the augmented program $\mathfrak{P}$.

Theorem 6. *For any program P, a conjunction of literals D is a non-hypothetical WnP justification of some q-literal L, i.e. $D \leq Why_P(L)$ iff there is a justification E, i.e. $E \leq \mathbb{W}_P(L)$ s. t. $\lambda^p(E)$ is the result of removing labels of the form 'not(A)' from D.* □

Theorem 6 establishes a correspondence between non-hypothetical WnP-justifications and (flattened) ECJ justifications. In the case of hypothetical justifications, they are not directly captured by ECJ, but can be obtained using the augmented program $\mathfrak{P}$ as stated by Theorem 5. As a byproduct we establish a formal relation between WnP and CG.

Theorem 7. *Let P be a program and D be a non-hypothetical and enabled WnP-justification of some atom A, i.e. $D \leq Why_P(A)$. Then, for all CG stable model $\tilde{I}$ of P, there is some CG-justification G w.r.t. $\tilde{I}$ s.t. D contains all the vertices of G.* □

5 Related Work

There exists a vast literature on causal reasoning in Artificial Intelligence (AI). Papers on reasoning about actions and change [18–20] 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* [21] (See [22] for an updated version). In this approach, it is possible to conclude cause-effect relations like "*A* has been an actual cause of *B*" from the behaviour of structural equations by applying, under some *contingency* (an alternative model in which some values are fixed) the *counterfactual dependence* interpretation from [23]: "had *A* not happened, *B* would not have happened." Causal networks and ECJ differ in their final goals. While the former focuses on revealing a *unique* everyday-concept of causation (*actual causation*), the latter tries to provide precise definitions of *different concepts of causation*, leaving the choice of which concept corresponds to a particular scenario to the programmer. The approach of *actual causation* has also been followed in LP by [24] and [25].

As has been slightly discussed in the introduction, ECJ is also related to work by Hall [14, 15], who has emphasized the difference between two types of causal relations: *dependence* and *production*. The former relies on the idea "that counterfactual dependence between wholly distinct events is sufficient for causation." The latter is characterised by being *transitive*, *intrinsic* (two processes following the same laws must be both or neither causal) and *local* (causes must be connected to their effects via sequences of causal intermediates). In this sense, WnP is more oriented to *dependence* while CG is mostly related to *production*. ECJ is a combination of the dependence-oriented approach from WnP and the production-oriented behaviour from CG.

Focusing on LP, our work obviously relates to explanations obtained from ASP debugging approaches [5–11]. The most important difference of these works with respect to ECJ, and also WnP and CG, is that the last three provide fully algebraic semantics in which justifications are embedded into program models. A formal relation between [11] and WnP was established in [26] and so, using Theorems 5 and 6, it can be directly extended to ECJ, but at the cost of flattening the graph information (i.e. losing the order among rules).

6 Conclusions

In this paper we have introduced a unifying approach that combines causal production with enablers and inhibitors. We formally capture inhibited justifications by introducing a "non-classical" negation '$\sim$' in the algebra of causal graphs (CG). A inhibited justification is nothing else but an expression containing some negated label. We have also distinguished productive causes from enabling conditions (counterfactual dependences

that are not productive causes) by using a double negation '$\sim\sim$' for the latter. The existence of enabled justifications is a sufficient and necessary condition for the truth of a literal. Furthermore, our justifications capture, under the Well-founded semantics, both Causal Graph and Why-not Provenance justifications. As a byproduct we established a formal relation between these two approaches.

Using an example, we have also shown how to capture causal knowledge in the presence of dynamic defaults – those whose behaviour are not predetermined, but rely on some program condition – as for instance the inertia axioms. As pointed out by [27], causal knowledge is structured by a combination of *inertial laws* – how the world would evolve if nothing intervened – and *deviations* from these inertial laws. The importance of default knowledge has been widely recognised as a cornerstone of the problem of actual causation in [28, 29] among others.

Interesting issues for future study are incorporating enabled and inhibited justifications to the stable model semantics and replacing the syntactic definition in favour of a logical treatment of default negation, as done for instance with the Equilibrium Logic [30] characterisation of stable models. Other natural steps would be the consideration of syntactic operators for capturing more specific knowledge about causal information, like the influence of a particular event or label in a conclusion, and the representation of non-deterministic causal laws, by means of disjunctive programs and the incorporation of probabilistic knowledge. From a KR point of view, another interesting future line of study is to apply our semantics to other traditional examples of the actual causation literature like *short-circuits* and *switches* [15].

Acknowledgements We are thankful to Carlos Damasio for his suggestions and comments on earlier versions of this work. We also thank the anonymous reviewers for their help to improve the paper. This research was partially supported by Spanish Project TIN2013-42149-P.

References

1. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Ann. Math. Artif. Intell. **25**(3-4) (1999) 241–273
2. Marek, V., Truszczyński, M. Artificial Intelligence. In: Stable Models and an Alternative Logic Programming Paradigm. Springer Berlin Heidelberg (1999) 375–398
3. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Logic Programming: Proc. of the Fifth International Conference and Symposium (Volume 2). (1988)
4. Van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM (JACM) **38**(3) (1991) 619–649
5. Specht, G.: Generating explanation trees even for negations in deductive database systems. LPE **1993** (1993) 8–13
6. Pemmasani, G., Guo, H.F., Dong, Y., Ramakrishnan, C., Ramakrishnan, I.: Online justification for tabled logic programs. In: Functional and Logic Programming. Springer (2004)
7. Gebser, M., Pührer, J., Schaub, T., Tompits, H.: Meta-programming technique for debugging answer-set programs. In: Proc. of the 23rd Conf. on Artificial Inteligence (AAAI'08). (2008)
8. Oetsch, J., Pührer, J., Tompits, H.: Catching the ouroboros: On debugging non-ground answer-set programs. Theory and Practice of Logic Programming **10**(4-6) (2010) 513–529
9. Schulz, C., Toni, F.: Aba-based answer set justification. TPLP **13**(4-5-Online-Suppl.) (2013)

10. Denecker, M., De Schreye, D.: Justification semantics: A unifiying framework for the semantics of logic programs. In: Proc. of the LPNMR Workshop. (1993)
11. Pontelli, E., Son, T.C., El-Khatib, O.: Justifications for logic programs under answer set semantics. Theory and Practice of Logic Programming (TPLP) **9**(1) (2009) 1–56
12. Damásio, C.V., Analyti, A., Antoniou, G.: Justifications for logic programming. In: Proc. of LPNMR'13). (2013)
13. Cabalar, P., Fandinno, J., Fink, M.: Causal graph justifications of logic programs. TPLP **14**(4-5) (2014) 603–618
14. Hall, N.: Two concepts of causation. In Collins, J., Hall, E.J., Paul, L.A., eds.: Causation and counterfactuals. Cambridge, MA: MIT Press (2004) 225–276
15. Hall, N.: Structural equations and causation. Philosophical Studies **132**(1) (2007) 109–136
16. Cabalar, P., Fandinno, J., Fink, M.: A complexity assessment for queries involving sufficient and necessary causes. In: Logics in Artificial Intelligence. Volume 8761. (2014) 297–310
17. Van Gelder, A.: The alternating fixpoint of logic programs with negation. In: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, ACM (1989) 1–10
18. Lin, F.: Embracing causality in specifying the indirect effects of actions. In Mellish, C.S., ed.: Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI), Montreal, Canada, Morgan Kaufmann (August 1995)
19. McCain, N., Turner, H.: Causal theories of action and change. In: Proc. of the AAAI-97. (1997) 460–465
20. Thielscher, M.: Ramification and causality. Artificial Intelligence Journal **1-2**(89) (1997) 317–364
21. Pearl, J.: Causality: models, reasoning, and inference. Cambridge University Press, New York, NY, USA (2000)
22. Halpern, J.Y., Hitchcock, C.: Actual causation and the art of modeling. (2011)
23. Hume, D.: An enquiry concerning human understanding (1748) Reprinted by Open Court Press, LaSalle, IL, 1958.
24. Meliou, A., Gatterbauer, W., Halpern, J.Y., Koch, C., Moore, K.F., Suciu, D.: Causality in databases. IEEE Data Eng. Bull. **33**(EPFL-ARTICLE-165841) (2010) 59–67
25. Vennekens, J.: Actual causation in CP-logic. TPLP **11**(4-5) (2011) 647–662
26. Damásio, C.V., Analyti, A., Antoniou, G.: Justifications for logic programming. In: Proc. of LPNMR'13. (2013)
27. Maudlin, T.: Causation, counterfactuals, and the third factor. In Collins, J., Hall, E.J., Paul, L.A., eds.: Causation and Counterfactuals. MIT Press (2004)
28. Halpern, J.Y.: Defaults and normality in causal structures. In Brewka, G., Lang, J., eds.: Proc. of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), AAAI Press (2008) 198–208
29. Hitchcock, C., Knobe, J.: Cause and norm. Journal of Philosophy **11** (2009) 587–612
30. Pearce, D.: A new logical characterisation of stable models and answer sets. In: Non monotonic extensions of logic programming. Proc. NMELP'96. (LNAI 1216). (1996)

7 Proofs

Proposition 1. *Negation '$\sim$' is anti-monotonic. That is, given two terms t and u, then $t \leq u$ implies that $\sim t \geq \sim u$.* □

Proof. Note that by definition $t \leq u$ iff $t * u = t$ and then $\sim(t * u) = \sim t$ iff $\sim t + \sim u = \sim t$ iff $\sim t \geq \sim u$. □

Proposition 2. *Given any term t, it can be rewritten as an equivalent term u in negation normal form.* □

Proof. This is a trivial proof by structural induction using the DeMorgan laws and negation of application axiom. Furthermore, using the axiom $\sim\sim\sim t = t$ no more than two nested negations are required. □

Theorem 8 (From [12]). *Given a labelled logic program P, let N be a set of facts not in program P and R be a subset of rules of P. A literal L belongs to the WFM of $(P \backslash R) \cup N$ iff there is a conjunction of literals $D \models Why_P(L)$, such that $Remove(D) \subseteq R$, $Keep(D) \cap R = \emptyset$, $AddFacts(D) \subseteq N$, and $NoFacts(D) \cap N = \emptyset$.* □

Proof of Theorem 1 . Form Theorem 8, a literal L holds with respect to $\mathbb{W}_P$ if and only there is a positive WnP justification $D \leq Why_P(L)$. Positive WnP justifications are non-hypothetical and enabled. Thus, whether there is a is a positive WnP-justification $D \leq Why_P(L)$, then, by Theorem 6, there is a enabled justification $E \leq \mathbb{W}_P(L)$. The other way around, if there is a enabled justification $E \leq \mathbb{W}_P(L)$, also by Theorem 6, there is a positive WnP-justification $D \leq Why_P(L)$. That is, there is a positive WnP-justification $D \leq Why_P(L)$ if and only if there is a enabled justification $E \leq \mathbb{W}_P(L)$. Therefore a q-literal L holds with respect to $\mathbb{W}_P$ if and only if there is some enabled justification E of L. □

Definition 13 (Direct consequences). *Given a positive logic program P over signature $\langle At, Lb \rangle$, the operator of* direct consequences *is a function T_P from interpretations to interpretations such that*

$$T_P(I)(A) \stackrel{\text{def}}{=} \sum \Big\{ \big(I(B_1) * \ldots * I(B_n) \big) \cdot t \;\Big|\; (t : p \leftarrow B_1, \ldots, B_n) \in P \Big\}$$

for any interpretation I and any atom $A \in At$.

Theorem 9 (From [13]). *Let P be a (possibly infinite) positive logic program. Then, (i) $lfp(T_P)$ is the least model of P, (ii) furthermore, if P is positive and has n rules, then $lfp(T_P) = T_P{\uparrow}^{\omega}(\mathbf{0}) = T_P{\uparrow}^{n}(\mathbf{0})$.* □

Proposition 3. *Let P be a (possibly infinite) positive logic program. Then, (i) $lfp(T_P)$ is the least model of P, (ii) furthermore, if P is positive and has n rules, then $lfp(T_P) = T_P{\uparrow}^{\omega}(\mathbf{0}) = T_P{\uparrow}^{n}(\mathbf{0})$.* □

Proof. Each B_i is either an atom or a term. By assuming that those B_i that are terms are in negation normal form and treating each term of the form $\sim l$ and $\sim\sim l$ as new labels it follows that every ECJ program is a CG program. Hence, by Theorem 9, it follows that $lfp(T_P) = T_P{\uparrow}^{\omega}(\mathbf{0}) = T_P{\uparrow}^{n}(\mathbf{0})$. Finally, note that ECJ satisfies the same equalities thatn CG (and more), so $t = u$ under CG implies $t = u$ under ECJ. □

Lemma 1. *Let P_1 and P_2 be two positive programs and U_1 and U_2 be two interpretations such that $P_1 \supseteq P_2$ and $U_1 \leq U_2$. Let also I_1 and I_1 be the least models of programs ${P_1}^{U_1}$ and ${P_2}^{U_2}$, respectively. Then $I_1 \geq I_2$.* □

Proof. Firstly, for all rule r_i and pair of interpretations J_1 and J_2 s.t. $J_1 \geq J_2$,

$$J_1(body^+(r_i^{U_1})) \geq J_2(body(r_i^{U_2}))$$

Furthermore, since $U_1 \leq U_2$, by Proposition 1, it follows

$$U_1(body^-(r_i^{U_1})) \geq U_2(body^-(r_i^{U_2}))$$

Then, by Definition 4,

$$J_j(body^-(r_i^{U_1})) \stackrel{\text{def}}{=} U_j(body^-(r_i^{U_1}))$$

and then

$$J_1(body^-(r_i^{U_1})) \geq J_2(body^-(r_i^{U_2}))$$

Hence, we obtain that $J_1(body(r_i^{U_1})) \geq J_2(body(r_i^{U_2}))$.

Since $P_1 \supseteq P_2$, it follows that every rule $r_i \in P_2$ is in P_1 as well and then it holds that $T_{{P_1}^{U_1}}(I)(p) \geq T_{{P_2}^{U_2}}(I)(p)$ for all atom A. Furthermore, since

$$T_{{P_1}^{U_1}} \uparrow^0 (\mathbf{0})(A) = T_{{P_2}^{U_2}} \uparrow^0 (\mathbf{0})(A) = 0$$

it follows $T_{{P_1}^{U_1}} \uparrow^i (\mathbf{0})(A) \geq T_{{P_1}^{U_1}} \uparrow^i (\mathbf{0})(A)$ for all $0 \leq i$. Finally,

$$T_{{P_j}^{U_j}} \uparrow^\omega (\mathbf{0})(A) \stackrel{\text{def}}{=} \sum_{i \leq \omega} T_{{P_j}^{U_j}} \uparrow^i (\mathbf{0})(A) = 0$$

and hence $T_{{P_1}^{U_1}} \uparrow^\omega (\mathbf{0})(A) \geq T_{{P_1}^{U_1}} \uparrow^\omega (\mathbf{0})(A)$. By Theorem 9, these are respectively the least models of ${P_1}^{U_1}$ and ${P_2}^{U_2}$. That is $I_1 \geq I_2$. □

Proposition 4. *The Γ_P operator is anti-monotonic and the operator Γ_P^2 is monotonic. That is, $\Gamma_P(U_1) \geq \Gamma_P(U_2)$ and $\Gamma_P^2(U_1) \geq \Gamma_P^2(U_2)$ for any pair of interpretation U_1 and U_2 such that $U_1 \leq U_2$.* □

Proof. Since $U_1 \leq U_2$, by Lemma 1, it follows $I_1 \geq I_2$ with I_1 and I_2 being respectively the least models of P^{U_1} and P^{U_2}. Then, $\Gamma_P(U_1) = I_1$ and $\Gamma_P(U_2) = I_2$ and, consequently, $\Gamma_P(U_1) \geq \Gamma_P(U_2)$. Since Γ_P is anti-monotonic it follows that Γ_P^2 is monotonic. □

Lemma 2. *Let t be a term. Then $\lambda^p(\sim t) = \neg\lambda^p(t)$.* □

Proof. We proceed by structural induction assuming that t is in negated normal form. In case that $t = a$ is elementary, it follows that $\lambda^p(\sim a) = \neg a = \neg\lambda^p(a)$. In case that $t = \sim a$ with a elementary, $\lambda^p(\sim t) = \lambda^p(\sim\sim a)$ and $\lambda^p(\sim\sim a) = a = \neg\neg a = \neg\lambda^p(\sim a) = \lambda^p(t)$. In case that $t = \sim\sim a$, with a elementary, $\lambda^p(\sim t) = \lambda^p(\sim\sim\sim a)$ and

$$\lambda^p(\sim\sim\sim a) = \lambda^p(\sim a) = \neg a = \neg\lambda^p(\sim\sim a) = \neg\lambda^p(t)$$

In case that $t = u + v$. Then

$$\lambda^p(\sim t) = \lambda^p(\sim u * \sim v) = \lambda^p(\sim u) \wedge \lambda^p(\sim v)$$

By induction hypothesis $\lambda^p(\sim u) = \neg\lambda^p(u)$ and $\lambda^p(\sim v) = \neg\lambda^p(v)$ and, therefore, it holds that $\lambda^p(\sim t) = \neg\lambda^p(u) \wedge \neg\lambda^p(v)$. Thus, $\neg\lambda^p(t) = \neg(\lambda^p(u) \vee \lambda^p(v)) = \neg\lambda^p(u) \wedge \neg\lambda^p(v) = \lambda^p(\sim t)$.

In case that $t = u \otimes v$ with $\otimes \in \{*, \cdot\}$. Then $\lambda^p(\sim t) = \lambda^p(\sim u + \sim v) = \lambda^p(\sim u) \vee \lambda^p(\sim v)$ and by induction hypothesis $\lambda^p(\sim u) = \neg\lambda^p(u)$ and $\lambda^p(\sim v) = \neg\lambda^p(v)$. Consequently it holds that $\lambda^p(\sim t) = \neg\lambda^p(t)$. □

Lemma 3. *Let t be a term ϕ a provenance term. If $\phi \leq \lambda^p(t)$, then $\lambda^p(\sim t) \leq \neg\phi$ and if $\lambda^p(t) \leq \phi$, then $\neg\phi \leq \lambda^p(\sim t)$.* □

Proof. If $\phi \leq \lambda^p(t)$, then $\phi = \lambda^p(t) * \phi$ and then $\neg\phi = \neg\lambda^p(t) + \neg\phi$ and, by Lemma 2, it follows that $\neg\phi = \lambda^p(\sim t) + \neg\phi$. Hence $\lambda^p(\sim t) \leq \neg\phi$.

Furthermore if $\lambda^p(t) \leq \phi$, then $\phi = \lambda^p(t) + \phi$ and then $\neg\phi = \neg\lambda^p(t) * \neg\phi$ and, by Lemma 2, it follows that $\neg\phi = \lambda^p(\sim t) * \neg\phi$. Hence $\neg\phi \leq \lambda^p(\sim t)$. □

Proof of Theorem 2

Lemma 4. *Let P be a labelled logic program and Q be the result of removing all rules labelled by $t \in \{l, \sim l\}$ with l a label. Let I, J, U and V be four causal interpretations such that J and V are the result of removing $\sim t$ in the NNF of I and U, respectively. Then $T_{P^V}(I)$ is the result of the result of removing $\sim t$ in the NNF of $T_{P^V}(J)$.* □

Proof. Let E be an addend of the disjunctive normal form of $T_{P^U}(I)(A)$. Then

$$E \;=\; (E_{B_1} * \ldots * E_{B_m} * E_{C_1} * \ldots * E_{C_n}) \cdot r_i$$

where $r_i^U \in P^U$ is a rule of form of (5) and $E_{B_j} \leq I(B_j)$ and $E_{C_j} \leq \sim U(C_j)$ for each positive literal B_j and each negative literal *not* C_j in the body of r_i.

1. Assume that $\sim t$ does not occur in E. Then $\sim t$ does not occur in E_{B_j} nor E_{C_j}. By hypothesis, E_{B_j} and E_{C_j} are addends of the disjunctive normal form of $J(B_j)$ and $\sim V(C_j)$, respectively. Then E is an addend of the disjunctive normal form of $T_{Q^V}(J)(A)$.
2. Otherwise, $\sim t$ occurs in some E_{B_j} or E_{C_j}. By hypothesis all the result of removing $\sim t$ from E_{B_j} and E_{C_j} are addends of $J(B_j)$ and $V(C_j)$, respectively. Hence, the result of removing $\sim t$ in E is an addend of $T_{Q^V}(J)(A)$.

The other way around. Let E' be an addend of the disjunctive normal form of $T_{Q^V}(J)(A)$. Then

$$E' = (E'_{B_1} * \ldots * E'_{B_m} * E'_{C_1} * \ldots * E'_{C_n}) \cdot r'_i$$

with a rule in the form of (5) in Q^V and each E'_{B_j} and each E'_{C_j} being addends of the disjunctive normal form of $J(B_j)$ and $\sim V(C_j)$ for each positive literal B_j and each

negative literal *not* C_j in the body of (5), respectively. By hypothesis, for all E'_{B_j} and E'_{C_j}, there are addends E_{B_j} and E_{C_j} in the disjunctive normal form of $I(B_j)$ and $\sim U(C_j)$, and E'_{B_j} and E'_{C_j} are the result of removing $\sim t$ in E_{B_j} and E_{C_j}. Finally, note that a rule r'_i is also in P^U and, thus, E' is also an addend of the disjunctive normal form of $T_{P^U}(I)(A)$. □

Lemma 5. *Let P be a labelled logic program and Q be the result of removing all rules labelled by $t \in \{l, \sim l\}$ with l a label. Let U and V two causal interpretations such that V are the result of removing $\sim t$ in the NNF of U. Let I and J be the least models of P^V and Q^U,respectively. Then J is the result of the result of removing $\sim t$ in the NNF of I.* □

Proof. The proof follows by induction using Lemma 4 as induction step. □

Lemma 6. *Let P be a labelled logic program and Q be the result of removing all rules with some labelled by $t \in \{l, \sim l\}$ with l a label. Let U and V two causal interpretations such that V are the result of removing $\sim t$ in the NNF of U. Then $\Gamma_Q(J)$ is the result of the result of removing all addends containing $\sim t$ in the NNF of $\Gamma_P(I)$.* □

Proof. Note that, by definition $\Gamma_P(U)$ and $\Gamma_Q(V)$ are respectively the least model of the programs P^U and Q^V. Then the lemma statement follows directly from Lemma 5. □

Lemma 7. *Let P be a labelled logic program and Q be the result of removing all rules with some label $t \in \{l, \sim l\}$ with l a label. Let U and V two causal interpretations such that V are the result of removing $\sim t$ in the NNF of U. An interpretations J is a fixpoint of Γ_Q^2 if and only if there is a fixpoint I of Γ_P^2 such that J is the result of removing $\sim t$ in the NNF of I.* □

Proof. For any pair of interpretations I' and J', by Lemma 6, it follows that $\Gamma_P(I')$ is the result of removing all addends containing l in the disjunctive normal form of $\Gamma_Q(J')$. Then $\Gamma_P^2(I') = \Gamma_P(\Gamma_P(I'))$ is the result of removing all addends containing l in the disjunctive normal form of $\Gamma_Q^2(J') = \Gamma_Q(\Gamma_Q(J'))$.

For the only if direction, let $I \stackrel{\text{def}}{=} \Gamma_P^2 \uparrow^\infty (J)$. Then I is a fixpoint of Γ_P^2 and inductively applying that $\Gamma_P^2(I)$ is the result of removing all addends containing l in the disjunctive normal form of $\Gamma_Q^2(J) = J$ it follows that I is the result of removing all addends containing l in the disjunctive normal form of J. The if direction is symmetric. □

Proof of Theorem 2 . In case that $L = A$ is an atom then $E \leq \mathbb{W}_P(A)$ iff $E \leq \mathbb{L}_P(A)$ and $E \leq \mathbb{W}_Q(A)$ iff $E \leq \mathbb{L}_Q(A)$. By Lemma 7, $\mathbb{L}_P$ is the result of removing $\sim l$ in $\mathbb{L}_Q$. If $E \leq \mathbb{L}_P(A)$ then there is some $F \leq \mathbb{L}_Q(A)$ such that F is the result of removing $\sim l$ in E. In case that $L = not\ A$ then $E \leq \mathbb{W}_P(not\ A)$ iff $E \leq \sim\mathbb{U}_P(not\ A)$ and $E \leq \mathbb{W}_{\mathfrak{P}}(not\ A)$ iff $E \leq \sim\mathbb{U}_{\mathfrak{P}}(not\ A)$ and the proof is analogous. □

Proof of Theorem 3

Definition 14. *Given a program P, a* CG interpretation *is a mapping* $\tilde{I} : At \longrightarrow \mathbf{C}_{Lb}$ *assigning a causal term to each atom. We also define the operator* $\tilde{\Gamma}_P(\tilde{I})$ *to be the least model of the program* $P^{\tilde{I}}$. □

Lemma 8. *Let P be a program, U and* $\tilde{V}$ *respectively be a CP and a CG interpretation such that* $\tilde{V} \leq \lambda^c(\tilde{U})$. *Let also I and* $\tilde{J}$ *respectively be a CP and a CG interpretation such that* $\tilde{J} \leq I$. *Then* $\tilde{T}_{P^{\tilde{V}}}(\tilde{J}) \leq T_{P^U}(I)$. □

Proof. For all $E \leq \tilde{T}_{P^{\tilde{V}}}(\tilde{J})(p)$ there is a rule in $P^{\tilde{V}}$ in the form of

$$r_i : p \leftarrow B_1, \ldots, B_m$$

that corresponds to a rule in the form of (4) in P and furthermore $E \leq (E_{B_1} * \ldots * E_{B_m}) \cdot r_i$ with each $E_{B_j} \leq \tilde{J}(B_j)$ and $\tilde{V}(C_j) = 0$ for all B_j and C_j in $body(r_i)$. Hence there is a rule in P^U in the form of (5) and, by hypothesis, $E_{B_j} \leq I(B_j)$ for all B_j. Furthermore, clearly $\tilde{V}(C_j) = 0 \leq {\sim}U(C_j)$ for all C_j. Hence $E \leq (I(B_1) * \ldots * I(B_m) * {\sim}U(C_1) * \ldots * {\sim}U(C_m)) \cdot r_i \leq T_{P^U}(I)$. □

Lemma 9. *Let P be a labelled logic program, U and* $\tilde{V}$ *respectively be a CP and a CG interpretation such that* $\tilde{V}(p) \leq \lambda^c(U)$. *Let also I and* $\tilde{J}$ *respectively be the least model of programs* P^U *and* $P^{\tilde{V}}$. *Then* $\tilde{J} \leq I$. □

Proof. Since I and $\tilde{J}$ are the least models respectively of programs P^U and $P^{\tilde{V}}$ it follows that $I = T_{P^U} \uparrow^{\omega} (\mathbf{0})$ and $\tilde{J} = \tilde{T}_{P^{\tilde{V}}} \uparrow^{\omega} (\mathbf{0})$. We proceed by induction on the number of iterations k. When $k = 0$, it follows that $\tilde{T}_{P^{\tilde{V}}} \uparrow^0 (\mathbf{0})(p) = 0 \leq T_{P^U} \uparrow^0 (\mathbf{0})(p)$. When $0 < k < \omega$, we assume as induction hypothesis the statement holds for the case $k-1$. Then, by Lemma 8, it follows for the case k.

Finally, when $k = \omega$, for all $E \leq \tilde{T}_{P^{\tilde{V}}} \uparrow^{\omega} (\mathbf{0})(p))$ there is some i such that $E \leq \tilde{T}_{P^{\tilde{V}}} \uparrow^i (\mathbf{0})(p))$ and then, by induction hypothesis,

$$E \leq \lambda^c(\tilde{T}_{P^{\tilde{V}}} \uparrow^i (\mathbf{0})(p)) \leq \lambda^c(T_{P^U} \uparrow^{\omega} (\mathbf{0})(p))$$

Therefore $\tilde{J} \leq I$. □

Lemma 10. *Let P be a labelled logic program, I and* $\tilde{J}$ *respectively be a CP and a CG interpretation such such that* $\tilde{V}(p) \leq \lambda^c(U)$. *Then* $\tilde{\Gamma}_P(\tilde{J}) \leq \Gamma_P(I)$. □

Proof. By Lemma 8 it follows that $\tilde{J'} \leq I'$ with I' and $\tilde{J'}$ respectively the least models of P^I and P^J and by definition $\tilde{\Gamma}_P(\tilde{J}) = \tilde{J'}$ and $\Gamma_P(I) = I'$. Hence $\tilde{\Gamma}_P(\tilde{J}) \leq \Gamma_P(I)$. □

Lemma 11. *Let P be a labelled logic program, U and* $\tilde{V}$ *respectively be a CP and a CG interpretation such that* $\tilde{V} = \lambda^p(U)$. *Let also I and* $\tilde{J}$ *respectively be a CP and a CG interpretation such that* $\tilde{J} \geq \lambda^c(I)$. *Then* $\tilde{T}_{P^{\tilde{V}}}(\tilde{J}) = \lambda^c(T_{P^U}(I))$. □

Proof. For all explanation $F \leq T_{P^U}(\tilde{J})(p)$ there is a rule r_i^U in P^U in the form of (5) and so that a rule in the form of (4) in P and there are justifications $F_{B_j} \leq I(B_j)$ and $F_{C_j} \leq \sim U(C_j)$ for all B_j and C_j such that $F \leq (F_{B_1} * \ldots * F_{B_m} * F_{C_1} * \ldots * F_{C_n}) \cdot r_i$. If F_{C_j} contains a negative label for some C_j, then $\lambda^c(F_{C_j}) = 0$ and consequently it follows that $\lambda^c(F) = 0 \leq \tilde{T}_{P^{\tilde{V}}}(\tilde{J})(p)$. Thus, we assume that F_{C_j} does not contain negated labels, i.e. it only contains double negated labels, for all C_j. Therefore $\lambda^c(F_{C_j} = 1$ and there is not positive justification of C_j w.r.t. U for all atom C_j and so that $\tilde{V}(C_j) = 0$ for all C_j. Thus there is a rule in $P^{\tilde{V}}$ in the form of

$$r_i : p \leftarrow B_1, \ldots, B_m$$

By hypothesis, $\lambda^c(F_{B_j}) \leq \tilde{T}_{P^{\tilde{V}}}(B_j)$ for all B_j. Hence $\lambda^c(F) \leq \tilde{T}_{P^{\tilde{V}}}(p)$ for all atom p and, thus, $\tilde{T}_{P^{\tilde{V}}}(\tilde{J}) \geq \lambda^c(T_{P^U}(\tilde{I}))$. Furthermore, by Lemma 8, it follows that $\tilde{T}_{P^{\tilde{V}}}(\tilde{J}) \leq T_{P^U}(\tilde{I})$. Hence $\tilde{T}_{P^{\tilde{V}}}(\tilde{J}) = \lambda^c(T_{P^U}(\tilde{I}))$. □

Lemma 12. *Let P be a labelled logic program, U and $\tilde{V}$ respectively be a CP and a causal CG such that $\tilde{V} = \lambda^c(U)$ for all atom p. Let also I and $\tilde{J}$ respectively be the least model of programs P^U and $P^{\tilde{V}}$. Then $\tilde{J} = \lambda^c(I)$.* □

Proof. Since I and $\tilde{J}$ are the least models respectively of programs P^U and $P^{\tilde{V}}$ it follows that $I = T_{P^U} \uparrow^{\omega} (\mathbf{0})$ and $\tilde{J} = \tilde{T}_{P^{\tilde{V}}} \uparrow^{\omega} (\mathbf{0})$. We proceed by induction on the number of iterations k. When $k = 0$, it follows that $\tilde{T}_{P^{\tilde{V}}} \uparrow^0 (\mathbf{0})(p) = 0 \geq \lambda^c(0) = \lambda^c(T_{P^U} \uparrow^0 (\mathbf{0})(p))$. When $0 < k < \omega$, we assume as induction hypothesis the statement holds for the case $k-1$. Then, by Lemma 11, it follows for the case k.

Finally when $k = \omega$, for all $F \leq T_{P^U} \uparrow^{\omega} (\mathbf{0})(p))$ there is some i s.t. $F \leq T_{P^U} \uparrow^i (\mathbf{0})(p))$ and then, by induction hypothesis, $\lambda^c(F) \leq \tilde{T}_{P^{\tilde{V}}} \uparrow^i (\mathbf{0})(p)$ and, since furthermore it holds that $\tilde{T}_{P^{\tilde{V}}} \uparrow^i (\mathbf{0})(p)) \leq \tilde{T}_{P^{\tilde{V}}} \uparrow^{\omega} (\mathbf{0})(p)$, then $\lambda^c(F) \leq \tilde{T}_{P^{\tilde{V}}} \uparrow^{\omega} (\mathbf{0})(p)$. That is $\tilde{J} \geq \lambda^c(I)$. Furthermore, by Lemma 8, it follows that $\tilde{J} \leq I$. Hence $\tilde{J} = \lambda^c(I)$. □

Lemma 13. *Let P be a labelled logic program, I and $\tilde{J}$ respectively be a CP and a CG interpretation such that $\tilde{V} = \lambda^c(U)$. Then $\tilde{\Gamma}_P(\tilde{J}) \geq \lambda^c(\Gamma_P(I))$.* □

Proof. By Lemma 12 it follows that $\tilde{J'} = I'$ with I' and $\tilde{J'}$ respectively the least models of P^I and P^J and by definition $\tilde{\Gamma}_P(\tilde{J}) = \tilde{J'}$ and $\Gamma_P(I) = I'$. Hence $\tilde{\Gamma}_P(\tilde{J}) \geq \lambda^c(\Gamma_P(I))$. □

Proof of Theorem 3 . Let $\tilde{I}$ causal stable model of P and let $I \stackrel{\text{def}}{=} \Gamma_P^2 \uparrow^{\infty} (\tilde{I})$ be the least fixpoint of Γ_P^2 iterating on $\tilde{I}$. By Lemma 13, it follows that $\tilde{\Gamma}_P(\tilde{I}) = \lambda^c(\Gamma_P(\tilde{I}))$. Then, since $\tilde{I}$ is a causal stable model of P (i.e. $\tilde{I} = \tilde{\Gamma}_P(\tilde{I})$), it also holds that $\tilde{I} = \lambda^c(\Gamma_P(\tilde{I}))$. Furthermore, since $\tilde{I} = \lambda^c(\Gamma_P(\tilde{I}))$, it follows, again by Lemma 13, that

$$\tilde{I} = \tilde{\Gamma}_P(\tilde{I}) = \lambda^c(\Gamma_P(\Gamma_P(\tilde{I}))) = \lambda^c(\Gamma_P^2(\tilde{I})))$$

Inductively applying this argument, $\tilde{I} = \lambda^c(\Gamma_P^2 \uparrow^{\infty} (\tilde{I})) = \lambda^c(I) = \lambda^c(\tilde{\Gamma}_P(I))$.

The other way around. Let I be a fixpoint of Γ_P^2 such that $\lambda^c(I) = \lambda^c(\Gamma_P(I))$ and let $\tilde{I} \stackrel{\text{def}}{=} \lambda^c(I)$. Then , by Lemma 13, it follows that $\tilde{\Gamma}_P(\tilde{I}) = \lambda^c(\Gamma_P^2(I)) = \lambda^c(I) = \tilde{I}$. That is $\tilde{\Gamma}_P(\tilde{I}) = \tilde{I}$ and so that $\tilde{I}$ is a causal stable model of P. □

Proof of Theorem 4 . Let $\tilde{I}$ be a causal stable model of P and I be the correspondent fixpoint of Γ_P^2. Since E is a enabled justification of A, i.e. $E \leq \mathbb{W}_P(A)$, then $E \leq \mathbb{L}_P(A)$ with $\mathbb{L}_P$ the least fixpoint of Γ_P^2. Therefore $E \leq I(A)$ and, since E is enabled, $\lambda^c(I)(A) \neq 0$. Them, it follows that $\tilde{I}(A) = \lambda^c(I)$ and, so that, $\lambda^c(E) \leq \tilde{I}(A)$. Then $G \stackrel{\text{def}}{=} graph(\lambda^c(E))$ is, by definition, a causal explanation of the atom A.

Definition 15. *Given a program P, a* WnP interpretation *is a mapping* $\mathfrak{I} : At \longrightarrow \mathbf{B}_{Lb}$ *assigning a boolean formulas to each atom. We also define the operator* $\mathfrak{G}_{\mathfrak{P}}(\mathfrak{I})$ *to be the least model of the program* $P^{\mathfrak{I}}$. □

Lemma 14. *Let P be a labelled logic program, U and* $\mathfrak{U}$ *respectively be a CP and a WnP interpretation such that* $\mathfrak{U} \leq \lambda^p(U)$. *Let also I and* $\mathfrak{I}$ *respectively be a CP and WnP interpretation such that* $\lambda^p(\tilde{I}) \leq \mathfrak{I}$. *Then it holds that* $\lambda^p(T_{\mathfrak{P}^U}(\tilde{I})) \leq \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}(\mathfrak{I})$.

Proof. Suppose not, and let the atom A and the CP justification E be the witnesses. Since $E \leq T_{\mathfrak{P}^U}(\tilde{I})(A)$, there is a rule $r_i \in \mathfrak{P}$ in the form of (4) and

$$E \;=\; (E_{B_1} * \ldots * E_{B_1} * E_{C_1} * \ldots * E_{C_1}) \cdot r_i$$

where $E_{B_j} \leq I(B_j)$ and $E_{C_j} \leq {\sim}U(C_j)$. By hypothesis $\lambda^p(I) \leq \mathfrak{I}$ and $\mathfrak{U} \leq \lambda^p(U)$. Since $\lambda^p(I) \leq \tilde{\mathfrak{I}}$ and $E_{B_j} \leq I(B_j)$, it follows that, $\lambda^p(E_{B_j}) \leq \mathfrak{I}(B_j)$. Furthermore, since $\mathfrak{U} \leq \lambda^p(U)$, by Lemma 3, it follows that $\lambda^p({\sim}U) \leq \neg\mathfrak{U}$ and then, since $E_{C_j} \leq {\sim}U(C_j)$, it follows that $\lambda^p(E_{C_i}) \leq \neg\mathfrak{U}(C_j)$ for all C_j. Therefore $\lambda^p(E) \leq \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}(\mathfrak{I})(A)$ which is a contradiction with the assumption. □

Lemma 15. *Let P be a causal logic program, U and* $\mathfrak{U}$ *respectively be a CP and a WnP interpretation such that* $\lambda^p(U) \leq \mathfrak{U}$. *Let also I and* $\mathfrak{I}$ *respectively be a CP and WnP interpretation such that* $\mathfrak{I} \leq \lambda^p(I)$. *Then it holds that* $\mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}(\mathfrak{I}) \leq \lambda^p(T_{\mathfrak{P}^U}(I))$. □

Proof. Suppose not, and let the atom A and the WnP justification D be the witness. Since $D \leq \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}(I)(A)$, there is some rule $r_i \in \mathfrak{P}^{\mathfrak{U}}$ in the form of (4) and

$$D \;=\; D_{B_1} * \ldots * D_{B_m} * D_{C_1} * \ldots * D_{C_n} * r_i$$

where $D_{B_j} \leq \mathfrak{I}(B_j)$ for each B_j and $D_{C_j} \leq \neg\mathfrak{U}(C_j)$ for each C_j. By hypothesis, $\lambda^p(U) \leq \mathfrak{U}$ and $\mathfrak{I} \leq \lambda^p(I)$. Since $D_{B_j} \leq \mathfrak{I}(B_j)$ and $\mathfrak{I} \leq \lambda^p(I)$, it follows that $D_{B_j} \leq \lambda^p(I)(B_j)$. Furthermore, since $\lambda^p(U) \leq \mathfrak{U}$, by Lemma 3, it follows that $\neg\mathfrak{U} \leq \lambda^p({\sim}U)$ and then, since $D_{C_j} \leq \neg\mathfrak{U}(C_j)$, it follows that $D_{C_j} \leq \lambda^p({\sim}U)(C_j)$. Therefore $E \stackrel{\text{def}}{=} (E_{B_1} * \ldots * E_{B_1} * E_{C_1} * \ldots * E_{C_1}) \cdot r_i \leq T_{P^U}(I)(A)$ such that $D_{B_j} \leq \lambda^p(E_{B_j})$ and $D_{C_j} \leq \lambda^p(E_{C_j})$. Then $D \leq \lambda^p(E)$ which contradicts the assumption. □

Lemma 16. *Let P be a causal logic program, U and* $\mathfrak{U}$ *respectively be a CP and a WnP interpretation such that* $\lambda^p(U) = \mathfrak{U}$. *Let also I be a CP-interpretation. Then* $\mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}(\lambda^p(I)) = \lambda^p(T_{\mathfrak{P}^U}(I))$. □

Proof. The proof directly follows by combination of Lemmas 14 and 15. □

Lemma 17. *Let P be a causal logic program, U and* $\mathfrak{U}$ *respectively be a CP and a WnP interpretation such that* $\lambda^p(U) = \mathfrak{U}$. *Let also I be the least model of the program* $\mathfrak{P}^U$. *Then* $\lambda^p(I)$ *is the least model of* $\mathfrak{P}^{\mathfrak{U}}$. □

Proof. Since I and $\mathfrak{I}$ respectively are the least fixpoints of $T_{\mathfrak{P}^U}$ and $\mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}$, it follows, by Theorem 9, that $I = T_{\mathfrak{P}^U}\uparrow^{\omega}(\mathbf{0})$ and $\mathfrak{I} = \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}\uparrow^{\omega}(\bot)$. We proceed by induction on the number of iterations k. When $k = 0$, it follows that $\lambda^p(T_{\mathfrak{P}^U}\uparrow^0(\mathbf{0})(p)) = \lambda^p(0)$ and $\mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}\uparrow^0(\bot)(p) = 0$ and, since $\lambda^p(0) = \bot$, then it follows that

$$\lambda^p(T_{\mathfrak{P}^U}\uparrow^0(\mathbf{0})(p)) = \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}\uparrow^0(\bot)(p)$$

When $0 < k < \omega$, we assume as induction hypothesis the statement holds for the case $k-1$. Then, by Lemma 16, it follows for the case k. Finally when $k = \omega$, for all justification $E \leq T_{\mathfrak{P}^U}\uparrow^{\omega}(\mathbf{0})(p))$ (resp. for all $D \leq \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}\uparrow^{\omega}(\bot)(p))$) there is some $i < \omega$ s.t. $E \leq T_{\mathfrak{P}^U}\uparrow^{i}(\mathbf{0})(p))$ (resp. $D \leq \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}\uparrow^{i}(\bot)(p))$). By induction hypothesis, $\lambda^p(E) \leq \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}\uparrow^{i}(\bot)(p)$ (resp. $D \leq \lambda^p(T_{P^U}\uparrow^{i}(\mathbf{0})(p)))$). Therefore it holds that $\lambda^p(E) \leq \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}\uparrow^{\omega}(\bot)(p)$ (respectively $D \leq \lambda^p(T_{P^U}\uparrow^{\omega}(\mathbf{0})(p))$). Consequently, it holds that $\lambda^p(T_{\mathfrak{P}^U}\uparrow^{\omega}(\mathbf{0})(p)) = \mathfrak{T}_{\mathfrak{P}^{\mathfrak{U}}}\uparrow^{\omega}(\bot)(p)$. □

Lemma 18. *Let P be a labelled logic program and I be a CP interpretation. Then $\mathfrak{G}_{\mathfrak{P}}(\lambda^p(I)) = \lambda^p(\Gamma_{\mathfrak{P}}(I))$.* □

Proof. By definition $\Gamma_{\mathfrak{P}}(I)$ is the least models of $\mathfrak{P}^I$. Then, by Lemma 17, it follows that $\lambda^p(\Gamma_{\mathfrak{P}}(I))$ is the least model of $\mathfrak{P}^{\lambda^p(I)}$ therefore $\mathfrak{G}_{\mathfrak{P}}(\lambda^p(I)) = \lambda^p(\Gamma_{\mathfrak{P}}(I))$. □

Lemma 19. *Let P be a labelled logic program and I and J be two fixpoints of the operator Γ_P^2 such that $\lambda^p(J) = \lambda^p(I)$. Then $I = J$.* □

Proof. Since $\lambda^p(J) = \lambda^p(I)$, then $\neg\lambda^p(J) = \neg\lambda^p(I)$ and by Lemma 3, $\lambda^p(\sim J) = \lambda^p(\sim I)$, furthermore, this implies $\sim J = \sim I$. Hence $P^I = P^J$ and $\Gamma_P(I) = \Gamma_P(J)$. Then it is clear that $\Gamma_P^2(I) = \Gamma_P^2(J)$, i.e. $I = J$. □

Theorem 10. *Let P be a labelled logic program and $\mathfrak{I}$ be WnP interpretation. Then $\mathfrak{I}$ is a fixpoint of the operator $\mathfrak{G}_{\mathfrak{P}}^2$ if and only if there is a CP fixpoint I of Γ_P^2 such that $\mathfrak{I} = \lambda^p(I)$. Furthermore, given $\mathfrak{I}$, then I is unique.* □

Proof. By Lemma 18, for any interpretation J s.t. $J = \lambda^p(\mathfrak{I})$, $\mathfrak{G}_{\mathfrak{P}}(\lambda^p(J)) = \lambda^p(\Gamma_P(J))$. Then

$$\Gamma_P^2(\mathfrak{I}) = \lambda^p(\Gamma_P(\lambda^p(\Gamma_P(J))))$$

Since, for all J, it holds that $\Gamma_P(\lambda^p(J)) = \Gamma_P(J)$, then

$$\Gamma_P^2(\mathfrak{I}) = \lambda^p(\Gamma_P^2(J))$$

Hence, if $\mathfrak{I}$ is a fixpoint of Γ_P^2, i.e. $\Gamma_P^2(\mathfrak{I}) = \mathfrak{I}$

$$\mathfrak{I} = \lambda^p(\Gamma_P^2(J))$$

Therefore, for $I \stackrel{\text{def}}{=} \Gamma_P^2\uparrow^{\infty}(J)$, it holds that

$$\mathfrak{I} = \lambda^p(\Gamma_P^2\uparrow^{\infty}(J))$$

Furthermore, let J be fixpoint such that $I \neq J$ and $\lambda^p(J) = \mathfrak{I}$. By Lemma 19, it follows that $I = J$ which is a contradiction. So that I is unique. The other way around. Since $I = \Gamma_P^2(I)$, by Lemma 18, $\lambda^p(I) = \lambda^p(\Gamma_P^2(I)) = \mathfrak{G}_{\mathfrak{P}}^2(\lambda^p(I))$. □

Proof of Theorem 5. According to the definition of [12], $Why_P(A) \stackrel{\text{def}}{=} \mathfrak{T}_{\mathfrak{P}}(A)$ where $\mathfrak{T}_{\mathfrak{P}}$ is the least fixpoint of the operator $\mathfrak{G}^2_{\mathfrak{P}}$. Furthermore, by Theorem 10, there is a fixpoint I of $\Gamma^2_{\mathfrak{P}}$ such that $\mathfrak{T}_{\mathfrak{P}} = \lambda^p(I)$. Then $\mathfrak{T}_{\mathfrak{P}} \geq \lambda^p(\mathbb{L}_{\mathfrak{P}})$ where $\mathbb{L}_{\mathfrak{P}}$ is the least fixpoint of $\Gamma_{\mathfrak{P}}$. Also by Theorem 10, $\lambda^p(\mathbb{L}_{\mathfrak{P}})$ is a fixpoint of $\mathfrak{G}^2_{\mathfrak{P}}$ and so that $\mathfrak{T}_{\mathfrak{P}} \leq \lambda^p(\mathbb{L}_{\mathfrak{P}})$. Hence $\mathfrak{T}_{\mathfrak{P}} = \lambda^p(\mathbb{L}_{\mathfrak{P}})$. Furthermore by definition $\mathbb{W}_{\mathfrak{P}}(A) = \mathbb{L}_{\mathfrak{P}}(A)$. Therefore $Why_P(A) = \lambda^p(\mathbb{W}_{\mathfrak{P}})(A)$. The proofs of $Why_P(not\, A)$ and $Why_P(undef\, A)$ are analogous. □

Theorem 11. *Let P be program. A term with no sums E is a justification of some q-literal L, i.e. $E \leq \mathbb{W}_P(L)$ if and only if there is some non-hypothetical justification F with respect to the augmented program $\mathfrak{P}$, i.e. $F \leq \mathbb{W}_{\mathfrak{P}}(L)$ and E is the result of removing each occurrence in the form of $\sim\sim not(B)$ from F.* □

Proof. By definition P is the result of removing all rules labelled with $\sim not(B)$ in $\mathfrak{P}$. In case that $L = A$ is an atom then $E \leq \mathbb{W}_P(A)$ iff $E \leq \mathbb{L}_P(A)$ and $E \leq \mathbb{W}_Q(A)$ iff $E \leq \mathbb{L}_Q(A)$. By Lemma 7, $\mathbb{L}_P$ is the result of removing $\sim\sim not(B)$ in $\mathbb{L}_Q$. If $E \leq \mathbb{L}_P(A)$ then there is some $F \leq \mathbb{L}_Q(A)$ such that F is the result of removing $\sim\sim not(B)$ in E. In case that $L = not\, A$ then $E \leq \mathbb{W}_P(not\, A)$ iff $E \leq \sim\mathbb{U}_P(not\, A)$ and $E \leq \mathbb{W}_{\mathfrak{P}}(not\, A)$ iff $E \leq \sim\mathbb{U}_{\mathfrak{P}}(not\, A)$ and the proofs are analogous. Then, E is justification of literal L, i.e. $E \leq \mathbb{W}_P(L)$ if and only there is some justification F of L w.r.t. $\mathfrak{P}$ i.e. $F \leq \mathbb{W}_{\mathfrak{P}}(L)$ such that E is the result of removing each occurrence of $\sim\sim not(B)$ in F. □

Proof of Theorem 6. By definition $D \leq Why_P(L)$ if and only if $D \leq \lambda^p(\mathbb{W}_{\mathfrak{P}})(L)$ if and only if there is $F \leq \mathbb{W}_{\mathfrak{P}}(L)$ such that $D = \lambda^p(F)$. That is, if D is a non-hypothetical WnP-justification of L, then there is a non-hypothetical justification F of L w.r.t. $\mathfrak{P}$ such that $D = \lambda^p(F)$. From Theorem 11 there is some $E \leq \mathbb{W}_P(L)$ such that E is the result of removing each occurrence of the form $\sim\sim not(B)$ from F. Then $\lambda^p(E)$ is the result of removing each occurrence in the form of $not(B)$ from $\lambda^p(F) = D$.

The other way around. If $E \leq \mathbb{W}_P(L)$, from Theorem 11 there is some $F \leq Why_P(L)$ and E is the result of removing each occurrence in the form of $\sim\sim not(B)$ from F. Then $\lambda^p(E)$ is the result of of removing each occurrence in the form of $not(B)$ from $\lambda^p(F) = D$. □

Proof of Theorem 7. If D is positive, then it is also non-hypothetical and then, by Theorem 6, there is a non-preempted justification E, i.e. $E \leq \mathbb{W}_P(L)$ such that $\lambda^p(E)$ is the result of removing each occurrence in the form on $not(B)$ from D. Then, from Theorem 4, $G \stackrel{\text{def}}{=} graph(\lambda^c(E))$ is a causal explanation of A and D contains all the labels of E and so that all the vertices of G. □