A complexity assessment for queries involving sufficient and necessary causes[*]

Pedro Cabalar[1], Jorge Fandinno[1] and Michael Fink[2]

[1] Department of Computer Science
University of Corunna, Spain
{cabalar, jorge.fandino}@udc.es
[2] Vienna University of Technology,
Institute for Information Systems
Vienna, Austria
fink@kr.tuwien.ac.at

Abstract. In this work, we revisit a recently proposed multi-valued semantics for logic programs where each true atom in a stable model is associated with a set of expressions (or causal justifications) involving rule labels. For positive programs, these causal justifications correspond to the possible alternative proofs of the atom that further satisfy some kind of minimality or lack of redundancy. This information can be queried for different purposes such as debugging, program design, diagnosis or causal explanation. Unfortunately, in the worst case, the number of causal justifications for an atom can be exponential with respect to the program size, so that computing the complete causal model may become intractable in the general case. However, we may instead just be interested in querying whether some particular set of rules are involved in the atom derivation, either as a *sufficient cause* (they provide one of the alternative proofs) or as a *necessary cause* (they are mandatorily used in all proofs). In this paper, we formally define sufficient and necessary causation for this setting and provide precise complexity characterizations of the associated decision problems, showing that they remain within the first two levels of the polynomial hierarchy.

1 Introduction

An important challenge in Knowledge Representation (KR) and Reasoning is not only deriving conclusions from a given theory or knowledge base, but also providing *explanations* for their derivation. This is particularly interesting in KR areas related to causal reasoning. For instance, in diagnosis scenarios, when discrepancies between observations and predictions are found, we may be interested not only in exhibiting a set of malfunctioning components, but also the way in which these breakdowns have eventually caused each discrepancy. Another example is legal reasoning, where determining a legal responsibility usually involves finding out which agent (or agents) have eventually caused a given result – checking whether the agent is involved in the explanation for

[*] This research was partially supported by Spanish MEC project TIN2009-14562-C05-04, by Xunta de Galicia, Spain, grant GPC2014/070 and program INCITE 2011, Inditex-University of Corunna 2013 grants, as well as by the Austrian Science Fund (FWF) project P24090.

that result is as important as the result occurrence itself. There are, however, different degrees in which a set of events or actions A may be "involved" in the explanation for some effect B. In some cases, A may *suffice* to explain B. In other cases, A alone cannot guarantee B, but is indispensable in any explanation for the latter, i.e., it is *necessary* for B. Let us illustrate these ideas with an example.

Example 1. An *alarm* is connected to three switches as depicted in Figure **??**. Each switch is operated by a different person and, at a given moment, they all accidentally close the switches. We want to analyse the responsibility for firing a false alarm.

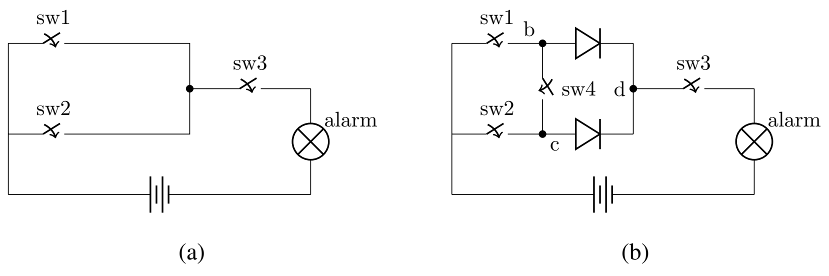

Fig. 1. A pair of circuits connecting switches and an alarm.

Analysing the circuit, we find two explanations for the alarm: moving down sw_1 and sw_3 together *suffices* to fire the alarm, and the same happens for sw_2 and sw_3. However, had sw_3 not been moved down, the alarm would have not been fired. That means that closing sw_3 is a *necessary cause* to fire the alarm, pointing out that the operator for that switch has, somehow, a higher degree of responsibility. Consider now the elaboration depicted in Figure **??** with a fourth switch and its corresponding person in charge, and suppose again that all persons close their respective switches. The set of events $\{sw_1, sw_3, sw_4\}$ obviously suffice to fire the alarm, since $\{sw_1, sw_3\}$ are still sufficient for that purpose. However, sw_4 is irrelevant, and so, it does not constitute an *actual cause*, whereas $\{sw_1, sw_3\}$ is a *sufficient cause* since nothing can be removed from it without ceasing to be a sufficient explanation.

Until now, we have made explanations in terms of actions, ignoring their connection to their effects through chains of intermediate events. Suppose that we want to reflect, for instance, the causal relation between the switch movements and the facts representing that there is current at wire points b, c or d in Figure **??**. To this aim, we will need to represent each explanation not just as a set of events, but as an ordered arrangement of them instead. For instance, the final effect for sw_1 is that the current reaches point d and the complete explanation for that effect would be now the sequence $sw_1 \cdot b \cdot d$. This, together with the action of closing sw_3, is a sufficient cause for *alarm*. Similarly, the joint occurrence of $sw_2 \cdot c \cdot d$ and sw_3 constitutes a second, alternative sufficient cause. A useful way of depicting explanations is by means of directed graphs with vertices representing events and edges representing causal connections among them. Figure **??** shows three sufficient explanations G_1, G_2 and G_3 for *alarm* corresponding to the circuit in Figure **??**. The first two explanations G_1 and G_2 are sufficient causes, whereas G_3 is not a cause, since we can "remove" sw_4 and b and still get the sufficient explanation G_2.

In this paper, we provide a formal definition for the three[3] different types of causal relations introduced above, that is *sufficient explanation*, *sufficient cause* and *necessary cause*, and study how these causal assertions can be derived from a representation in the form of a labelled logic program. To this aim, we use a recently proposed causal approach [**?**] that provides a multi-valued extension of the stable model semantics [**?**]. In this approach, each true atom in a stable model is associated an expression involving rule labels, called its *causal justification*, that has a direct relation to sets of causal graphs as those in Figure **??**. We summarise our contributions as follows.

– We formally define the concepts of *sufficient explanation*, *sufficient cause* and *necessary cause* for some atom (cf. Section **??**).
– We show that the number of possible sufficient causes for an atom can be, in the worst case, exponential with respect to the program size. Despite this fact, proving exact complexity characterisations (cf. Section **??** and see Figure **??**) of the associated decision problems, we establish that sufficient queries are not harder than traditional (brave or cautious) reasoning tasks under the stable model semantics.

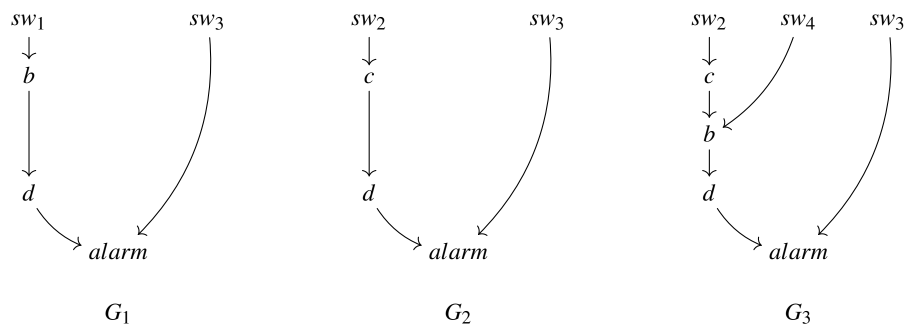

Fig. 2. Sufficient explanations for the alarm firing in Figure **??**. G_1 and G_2 are causes.

2 Background

In this section, we recall several definitions and notation from [**?**]. A *signature* is a pair $\langle At, Lb \rangle$ of sets that respectively represent *atoms* (or *propositions*) and rule *labels*.

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

[3] We leave the study of *actual causation* (that is, events that are needed for some sufficient cause) for future work.

	positive programs	with negation (brave)	with negation (cautions)
sufficient explanation	P	NP	coNP
sufficient cause	P	NP	coNP
necessary cause	coNP	Σ_2^P	coNP

Fig. 3. Completeness results for deciding different types of causation in causal logic programs.

Definition 1 (Causal logic program). *Given a signature $\langle At, Lb\rangle$, a* (causal) logic program *P is a set of rules of the form:*

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

where $t \in Lb \cup \{1\}$, H is an atom (the head*) and $B_1, \ldots, B_n$ are literals (the* body*).* □

For any rule R of the form (**??**) we define $label(R) \overset{\text{def}}{=} t$. We denote by $head(R) \overset{\text{def}}{=} H$ its *head*, and by $body(R) \overset{\text{def}}{=} \{B_1, \ldots, B_n\}$ 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.

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

Definition 2 (Causal term). *A* (causal) term*, t, over a set of labels Lb, is recursively defined as one of the following expressions $t ::= l \mid \prod S \mid \sum S \mid t_1 \cdot t_2 \mid (t_1)$ where $l \in Lb$, t_1, t_2 are in their turn causal terms and S is a (possibly empty and possible infinite) set of causal terms. When S is finite and non-empty, $S = \{t_1, \ldots, t_n\}$ we write $\prod S$ simply as $t_1 * \cdots * t_n$ and $\sum S$ as $t_1 + \cdots + t_n$. The set of causal terms is denoted by $\mathbf{T}_{Lb}$.* □

We assume that '$*$' has higher priority than '$+$'. When $S = \emptyset$, we denote, as usual $\prod S$ by 1 and $\sum S$ by 0. These values are the indentities for the product and the addition, respectively. All three operations, '$*$', '$+$' and '$\cdot$' are associative. Furthermore, '$*$' and '$+$' are commutative and they hold the usual absorption and distributive laws with respect to infinite sums and products of any completely distributive lattice, as shown[4] in Figure **??**. The behaviour of the '$\cdot$' operator is more specific from this approach and is captured by the properties shown in Figure **??**. Note that distributivity with respect to the product is applicable to terms c, d, e without sums (this means that the empty sum, 0, is not allowed either). As usual for lattices, we define an order relation $\leq$ as follows:

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

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

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

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

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

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

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

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

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

Fig. 5. Properties of the '$\cdot$' operator (c, d, e are terms without '$+$').

For interpretations I and J we say that $I \leq J$ whether $I(p) \leq J(p)$ for each atom $p \in At$. Hence, there is a $\leq$-bottom interpretation $\mathbf{0}$ (resp. a $\leq$-top interpretation $\mathbf{1}$) that maps each atom p to 0 (resp. 1). The value assigned to a negative literal *not p* by an interpretation I, denoted as $I(not\ p)$, is defined as: $I(not\ p) \stackrel{\text{def}}{=} 1$ if $I(p) = 0$; and $I(not\ p) \stackrel{\text{def}}{=} 0$ otherwise.

We define next a simple variation of the standard Gelfond and Lifschitz' program reduct [**?**]. The *reduct* of program P with respect to a 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 $B \in body(R)$; and (2) removing all negative literals from the remaining rules.

Definition 3 (Causal model). *Given a positive causal logic program P, a causal interpretation I is a* causal stable model*, in symbols $I \models P$, if and only if I is the $\leq$-least interpretation holding*

$$\big(\, I(B_1) * \ldots * I(B_n) \,\big) \cdot t \leq I(H)$$

for each rule $R \in P$ of the form (**??**)*. An interpretation I is a* causal stable model *of any program P iff I is a causal stable model of P^I.* □

Definition 4 (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 $p \in At$:*

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

Theorem 1 (From Theorem 2 in [?]). *Let P be a (possibly infinite) positive logic program with n causal rules. Then, (i) $lfp(T_P)$ is the least model of the program P, and (ii) $lfp(T_P) = T_P \uparrow^{\omega} (\mathbf{0}) = T_P \uparrow^{n} (\mathbf{0})$.* □

3 Query language

In order to characterise the different types of causation, we must begin first by a formal description of causal explanations. In particular, an explanation will have the form of a particular kind of graph involving rule labels, as defined below.

Definition 5 (Explanation or Causal graph). *Given a set of labels Lb, an* explanation *or* causal graph (c-graph) *G is a transitively and reflexively closed directed graph with a set of vertices $V \subseteq Lb$ and a set of edges $E \subseteq V \times V$. We denote the set of causal graphs by $\mathbf{C}_{Lb}$.* □

Imposing reflexivity is not essential, but is more convenient for obtaining simpler definitions. Transitivity, however, is crucial for defining an adequate ordering relation among explanations with the simple use of the subgraph relation. To see why, let us consider again the graphs G_2 and G_3 in Figure **??**. As we explained in the introduction, G_3 is a sufficient explanation for *alarm* but is not a sufficient *cause* because G_2 is also sufficient and somehow "smaller." In fact, G_2 can be obtained by "removing" b and sw_4 from G_3 while respecting the rest of causal dependence relations. However, G_2 is not a subgraph of G_3 since the edge (c,d) is not present in the latter. To capture this idea of being smaller as a result of "removing parts" we must use instead the transitive closures: the transitive closure of G_2 is indeed a subgraph of the transitive closure of G_3.

For any c-graph G we define an associated causal term $term(G)$ as follows:

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

Definition 6 (sufficient explanation, sufficient cause, necessary cause). *Given an interpretation I and an atom p we say that a c-graph G is*

- *a* sufficient explanation *for p iff $term(G) \leq I(p)$*
- *a* sufficient cause *of p iff it is a subgraph-minimal sufficient explanation for p*
- *a* necessary cause *of p iff it is a subgraph of all sufficient causes of p and $I(p) \neq 0$.* □

Example 2 (Ex. **??** *continued).* A possible representation of the circuit in Figure **??** is the logic program P_1 containing the following causal rules:

$$\begin{array}{rcl@{\qquad}rcl}
alarm: \ alarm(T) & \leftarrow & down(sw_3,T),\ current(c,T) & d: current(d,T) & \leftarrow & current(b,T) \\
b: current(b,T) & \leftarrow & down(sw_1,T) & d: current(d,T) & \leftarrow & current(c,T) \\
c: current(c,T) & \leftarrow & down(sw_2,T) & down(X,T) & \leftarrow & m(X,d,T) \\
 & & & up(X,T) & \leftarrow & m(X,u,T)
\end{array}$$

plus the corresponding inertia rules for atoms *up* and *down* (we consider that the rest of the fluents are non-inertial, and so, false by default):

$$up(X,T{+}1) \leftarrow up(X,T), not\ down(X,T{+}1) \qquad down(X,T{+}1) \leftarrow down(X,T), not\ up(X,T{+}1)$$

where X is any switch number $X \in \{1,2,3,4\}$ and T is a natural number representing a time instant. Atoms $m(X,D,T)$ represent the action of moving switch X up 'u' or down 'd' at time instant T. Consider now a story where, initially, all switches are up, then sw_1

and sw_2 are closed in Situation 1, then sw_3 is closed at 3 and finally sw_4 closed at 4. The following set of facts, added to $P_{??}$, captures this scenario:

$$\begin{array}{llll} & up(sw_s, 0) & & \text{for } s \in \{1,2,3\} \\ sw_1 : m(sw_1, d, 1) & sw_2 : m(sw_2, d, 1) & sw_3 : m(sw_3, d, 3) & sw_4 : m(sw_4, d, 4) \end{array}$$

In the least model I of $P_{??}$, $I(alarm) = (sw_1 \cdot b \cdot d * sw_3) \cdot alarm + (sw_2 \cdot c \cdot d * sw_3) \cdot alarm$. The correspondence between the left and right operands in the addition above with c-graphs G_1 and G_2 in Figure **??** is easy to see. For instance $sw_1 \cdot b \cdot d$ corresponds to the left branch of G_1, sw_3 to the right one and *alarm* is its root. In fact, it can be shown, by successive application of algebraic equivalences in Figures **??** and **??**, that

$$term(G_1) = (sw_1 \cdot b \cdot d * sw_3) \cdot alarm \quad \text{and} \quad term(G_2) = (sw_2 \cdot c \cdot d * sw_3) \cdot alarm$$

In other words, $I(alarm) = term(G_1) + term(G_2)$ in the only causal stable model. Now, it is also easy to see, by idempotence of addition, that $term(G_1) + I(alarm) = I(alarm)$ which implies that $term(G_1) \leq I(alarm)$. According to Definition **??**, this means that G_1 is, as we mentioned in the introduction, a sufficient explanation for *alarm*. Furthermore, no subgraph of G_1 is a sufficient explanation for p and consequently G_1 is also a sufficient cause of p. By a similar observation, G_2 is also a sufficient cause of p and it can be checked that, apart from G_1 and G_2, no other c-graph is a sufficient cause of p. $\square$

In the previous example, the causal term for *alarm* obtained in the unique stable model of the program was equal to the sum of all terms associated with its sufficient causes. In fact, this constitutes a general property, as stated below.

Theorem 2. *Given an interpretation I and an atom p, the following holds:*

- $I(p) = \sum \{ \, term(G) \mid G \text{ is a sufficient cause of } p \, \}$, *and*
- *any c-graph G is a necessary cause of p (Def.* **??***) iff* $I(p) \leq term(G)$ *and* $I(p) \neq 0$. $\square$

Finally, for a program with negation and its possible stable models, we define, as usual, cautious and brave versions of the three types of explanations defined before.

Definition 7. *Given a causal logic program P, an explanation of any type (sufficient explanation, sufficient cause or necessary cause) for an atom p is further said to be* brave *(resp.* cautious*) if it constitutes an explanation of that same type for p in some (resp. every) stable model of P.* $\square$

4 Complexity assessment

The table in Figure **??** summarizes our complexity assessment (completeness results). Each row represents a query type – sufficient explanation, sufficient cause and necessary cause. The first column contains results for positive programs (unique stable model), whereas the second and the third columns respectively show the results for brave and cautions reasoning for programs with negation. Note that for sufficient queries the complexity is the same as for checking the truth of an atom in standard stable model semantics. Subesequently, we establish these results formally, starting with membership.

In order to check whether a c-graph G is, for instance, a brave (resp. cautious) sufficient cause of a given atom p for a program P we can begin computing the standard (non-causal) stable models of P. Since the causal reduct removes negations depending on whether negated atoms are 0 or different from 0 and there exists a one-to-one correspondence between causal stable models and standard stable models (see [**?**] for more details), we can build the reduct P^J using each non-causal stable model J and then proceed to compute its least causal model iterating the direct consequences operator for that reduct, T_{P^J}. Due to [**?**, Theorem 6], there is a 1-to-1 correspondence between least causal models obtained in this way and causal stable models of the program. Now it would remain to check whether $term(G) \leq I(p)$ in some (resp. every) causal stable model I. Unfortunately, comparing two arbitrary causal terms t and t' is not an easy task (in fact it is coNP-hard). A naive approach for that comparison would be rewriting t and t' in a normal form where '$\cdot$' and products are not in the scope of additions, something that can be always achieved by applying distributive laws of '$\cdot$' and '$*$' with respect to '$+$'. Once in that normal form, comparison is more or less straightforward ($\Sigma t_i \leq \Sigma t'_j$ iff for each term t_i there is some $t'_j \geq t_i$, and comparing terms just containing products and '$\cdot$' is a simple task). However, applying distributivity may easily blow up complexity. Consider the positive program P_2 consisting of the rules:

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

It is easy to see that the interpretations of atoms p_1 and q_1 in the least causal model I of $P_{\mathbf{??}}$ are $a+b$ and $c+d$, respectively. The interpretation for p_2 corresponds to:

$$\begin{aligned} I(p_2) &= (I(p_1) * I(q_1)) \cdot m_2 = ((a+b)*(c+d)) \cdot m_2 \\ &= (a*c)\cdot m_2 \ + \ (a*d)\cdot m_2 \ + \ (b*c)\cdot m_2 \ + \ (b*d)\cdot m_2 \end{aligned}$$

This addition cannot be further simplified. Thus, by Theorem **??**, the four summands above are sufficient causes for p_2. Analogously, $I(q_2)$ can also be expressed as a sum of four sufficient causes – we just replace m_2 by n_2 in $I(p_2)$. But then, $I(p_3)$ corresponds to $(I(p_2) * I(q_2)) \cdot m_3$ and, applying distributivity, this yields a sum of 4×4 sufficient causes. In the general case, each atom p_n or q_n has $2^{2^{n-1}}$ sufficient causes so that expanding the complete causal value into this additive normal form becomes intractable.

Program $P_{\mathbf{??}}$ also reveals another issue. Even if distributivity is not applied, the causal terms directly obtained by the T_{P^J} operator for p_2 and q_2 require 4 operators, the causal terms for p_3 and q_3 require 10, $I(p_3) = ((a+b)*(c+d))\cdot m_2 * ((a+b)*(c+d))\cdot n_2\big) \cdot m_3$ and, in general, the terms for p_n or q_n would require $2^n + 2^{n-1} - 2$ operators. However, an interesting observation is that subterm $(a_1+b_1)*(c_1+d_1)$ occurs twice in $I(p_3)$ above, and the same happens for $I(q_3)$. This subterm will occur four times in the causal terms for atoms p_4 and q_4. Avoiding repetitions will allow us computing the least model of T_{P^J} in polynomial time (and thus, using a polynomial number of operators to represent it).

Definition 8 (Term and interpretation graph). *Given a set of labels Lb, a* term graph *(t-graph)* $\tilde{T} = \langle V, E, f_V, f_E, v_r \rangle$ *is a rooted, connected and labelled directed graph with*

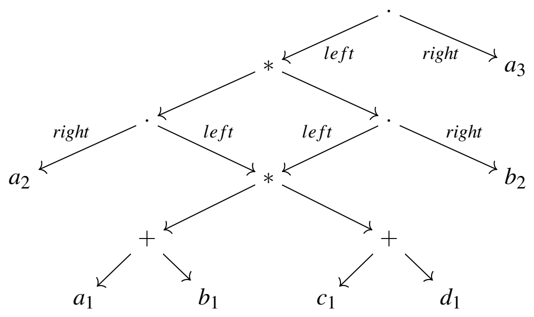

Fig. 6. The t-graph associated to $I(p_3)$ in program $P_{??}$.

a set of vertices V, edges E, root $v_r \in V$ and label functions $f_V : V \longrightarrow Lb \cup \{1, +, *, \cdot\}$ and $f_E : E \longrightarrow \{left, right\}$ such that

1. *all leafs are labelled with* unitary causes *(a label in Lb or 1),*
2. *all non-leaf nodes are labelled with operators*
3. *for any vertex labelled with the application operator '$\cdot$' there are exactly two outgoing edges labelled '$left$' and '$right$' being the target for the latter a leaf node. The rest of edges in the graph are unlabelled.* □

Each vertex in a t-graph $\tilde{T}$ represents a corresponding causal term as follows:

$$\begin{array}{ll} term_{\tilde{T}}(v) = f_V(v) & \text{for any leaf } v \text{ in } \tilde{T} \\ term_{\tilde{T}}(v) \overset{\text{def}}{=} \sum\{\ term(v') \ \big|\ (v,v') \in E\ \} & \text{if } f_V(v) = + \\ term_{\tilde{T}}(v) \overset{\text{def}}{=} \prod\{\ term(v') \ \big|\ (v,v') \in E\ \} & \text{if } f_V(v) = \ * \\ term_{\tilde{T}}(v) \overset{\text{def}}{=} term_{\tilde{T}}(u) \cdot term_{\tilde{T}}(w) & \text{if } f_V(v) = \ \cdot \ \text{ and } f_E(v,u) = left \\ & \qquad\qquad\quad \text{and } f_E(v,w) = right \end{array}$$

The term associated to a t-graph $\tilde{T}$ is the term associated with its root vertex v_r. As an example, Figure **??** represents the t-graph corresponding to $I(p_3)$ in the last example. We also extend these notions to interpretations. For an interpretation I, a *term interpretation* (t-interpretation) $\tilde{I}$ is just a function mapping each atom $p \in A$ to a t-graph $\tilde{T}$ such that $I(p) \ = \ term(\tilde{I}(p))$ for any atom p. We can now compute the least model of a positive program by iterating, a new direct consequences operator $\tilde{T}_P$ defined as

$\tilde{T}_P(\tilde{I})(p) \stackrel{\text{def}}{=} \langle V_p, E_p, f_{V,p}, f_{E,p}, v_p \rangle$ where:

$$\begin{array}{lcll}
V_p & \stackrel{\text{def}}{=} & \bigcup\{\ V_{\tilde{I}(R)} \cup \{v_p\} & \big|\ R \in P,\ head(R) = p\ \} \\
E_p & \stackrel{\text{def}}{=} & \bigcup\{\ E_{\tilde{I}(R)} \cup \{(v_p, v_R)\} & \big|\ R \in P,\ head(R) = p\ \} \\
V_{\tilde{I}(R)} & \stackrel{\text{def}}{=} & \bigcup\{\ V_{\tilde{I}(q)} & \big|\ q \in body(R)\ \} \cup \{\ v_R, w_R, v_l\ \} \\
E_{\tilde{I}(R)} & \stackrel{\text{def}}{=} & \bigcup\{\ E_{\tilde{I}(q)} \cup \{(w_R, v_{\tilde{I}(q)})\} & \big|\ q \in body(R)\ \} \cup \{\ (v_R, w_R)), (v_R, w_l)\ \}
\end{array}$$

$$f_{V,p}(v) \stackrel{\text{def}}{=} \begin{cases} + & \text{if } v = v_p \\ \cdot & \text{if } v = v_R \\ * & \text{if } v = w_R \\ label(R) & \text{if } v = w_l \\ f_{V,\tilde{I}(q)}(v) & \text{if } v \in V_{\tilde{I}(q)} \end{cases} \qquad f_{E,p}(e) \stackrel{\text{def}}{=} \begin{cases} left & \text{if } e = (v_R, w_l) \\ right & \text{if } e = (v_R, w_R) \\ f_{E,\tilde{I}(q)}(e) & \text{if } e \in E_{\tilde{I}(q)} \end{cases}$$

and, for any atom q, $V_{\tilde{I}(q)}$, $E_{\tilde{I}(q)}$, $f_{V,\tilde{I}(q)}$, $f_{E,\tilde{I}(q)}$, $v_{\tilde{I}(q)}$ are respectively the set of vertices, edges, the label functions of vertices and edges and the root of the t-graph $\tilde{I}(q)$.

Theorem 3. *Let P be a positive logic program with n rules, and let I be its least model. Then $I(p) = term(\tilde{T}_P \uparrow^n (\tilde{\mathbf{0}})(p))$ for all atoms p. Moreover, $\tilde{T}_P \uparrow^n (\tilde{\mathbf{0}})(p)$ is computable in polynomial time with respect to the size of P.* □

Theorem **??** builds on a polynomial time computable procedure to obtain the least model of a positive program P. We exploit now the fact that the term associated to a c-graph has no sums to define a boolean function $\mathit{sufficient}(G, \tilde{T}, v, l)$ that can be recursively computed as follows:

$$\begin{array}{ll}
\bigvee\{\ \mathit{sufficient}(G, \tilde{T}, v_i, l)\ \big|\ (v, v_i) \in E\ \} & \text{if } f_V(v) = + \\
\bigwedge\{\ \mathit{sufficient}(G, \tilde{T}, v_i, l)\ \big|\ (v, v_i) \in E\ \} & \text{if } f_V(v) = * \\
\mathit{sufficient}(G, \tilde{T}, v_l, l) & \text{if } f_V(v) = \cdot,\ f_E(v, v_l) = left,\ f_E(v, v_r) = right, \\
& \qquad \text{and } f_V(v_r) = 1 \\
\mathit{sufficient}(G, \tilde{T}, v_l, l_r) & \text{if } f_V(v) = \cdot,\ f_E(v, v_l) = left,\ f_E(v, v_r) = right, \\
& \qquad f_V(v_r) = l_r \text{ and } (l_r, l) \in G \\
true & \text{if } f_V(v) = 1 \text{ and } l = 1 \\
true & \text{if } f_V(v) = 1,\ l \neq 1 \text{ and } (l, l) \in G \\
true & \text{if } f_V(v) = l' \in Lb,\ l = 1 \text{ and } (l', l') \in G \\
true & \text{if } f_V(v) = l' \in Lb,\ l \neq 1 \text{ and } (l', l) \in G \\
false & \text{otherwise}
\end{array}$$

Then, we define $\mathit{sufficient}(G, \tilde{T}) \stackrel{\text{def}}{=} \mathit{sufficient}(G, \tilde{T}, root(\tilde{T}), 1)$.

Theorem 4. *Given an interpretation I and a t-interpretation $\tilde{I}$, a causal graph G is a sufficient explanation for an atom p with respect to I iff $\mathit{sufficient}(G, \tilde{I}(p)) = true$.* □

Corollary 1. *Given a positive causal logic program P, a causal graph G, and an atom p, deciding whether G is a sufficient explanation for p with respect to its least model is feasible in polynomial time.* □

In order to decide whether a causal graph is a sufficient cause, we recall that the *transitive reduction* of a directed graph G is another graph that preserves the reachability relation of G with a minimal set of edges. Deciding whether a c-graph G is a sufficient cause of some atom p in the least model of a program P can be done by:

1. checking whether G is a sufficient explanation for p
2. computing the transitive reduction G^R of G
3. computing the set S^R of graphs obtained from G^R by removing one of its edges
4. computing the set S obtained from the transitive closures of all graphs in S^R
5. checking for every causal graph G' in S that it is not a sufficient explanantion for p.

Theorem 5. *Given a positive logic program P, a causal graph G, and an atom p, deciding whether G is a sufficient cause of p with respect to its least model is feasible in polynomial time.* □

We consider now sufficient explanation and sufficient cause queries for programs with negation using the following nondeterministic procedure:

1. guessing a set of atoms J
2. checking whether J is the least classical model of the reduct P^J (ignoring labels)
3. checking whether G is a sufficient explanation for (resp. cause of) p w.r.t. P^J

This will succeed for some (resp. all) sets J iff G is a sufficient explanation for/cause of p with respect to some (resp. all) causal stable model(s) of P. Consequently we have the following membership results for brave (resp. cautions) sufficient explanations/causes:

Theorem 6. *Given a program P, deciding whether a c-graph G is a brave (resp. cautious) sufficient explanation or sufficient cause of an atom p is in NP (resp. in coNP).* □

In a similar way, we can decide whether G is a necessary cause as follows:

1. guess a set of atoms J and a causal graph G'
2. check whether J is the least classical model of the reduct of P^J (ignoring labels)
3. succeed if J does not contain p
4. check whether G' is a sufficient cause of p w.r.t. P^J
5. check whether G is not a subgraph of G'

This procedure succeeds iff G is not a cautious necessary cause of p with respect to program P yielding the following result:

Theorem 7. *Given a causal logic program P, deciding whether a c-graph G is a cautious necessary cause of an atom p is in coNP.* □

Finally, we can check brave necessary causation for an atom p by:

1. non-deterministically guessing a set of atoms J,
2. checking whether J is the least classical model of the reduct P^J (ignoring labels)
3. checking with an NP-oracle (Theorem **??**) whether G is necessary for p w.r.t. P^J

This will succeed iff G is necessary for p with respect to some stable model of program P. Hence, the problem can be decided non-deterministically in polynomial time with an NP-oracle ($\text{NP}^{NP} = \Sigma^P_2$):

Theorem 8. *Given a causal logic program P, deciding whether a c-graph G is a brave necessary cause of an atom p is in Σ^P_2.* $\square$

Turning to hardness, first note that any standard logic program is also an unlabelled causal logic program and an atom p is true in the former iff the empty c-graph (causal term 1) is a sufficient explanation (resp. a sufficient cause) for p. Therfore, the following result trivially follows:

Theorem 9. *Given a causal logic program P, deciding whether a c-graph G is a brave (resp. cautious) sufficient explanation or sufficient cause of some atom p is NP-complete (resp coNP-complete). Furthermore, P-hardness holds when P is positive.* $\square$

Let us next turn to the complexity results for necessary cause decision problems and show that they are tight too. First, we show that deciding whether a c-graph G is a brave necessary cause of some atom is Σ^P_2-hard by constructing a log-space reduction of deciding the truth of any quantified boolean formula of form $\varphi = \exists y_1,\dots,y_n \forall x_1,\dots,x_m\ \rho$, where $\rho = \psi_1 \vee \ldots \vee \psi_r$ and each $\psi_i = L_{i1} \wedge L_{i2} \wedge L_{i3}$ is a conjunction of three literals L_{ij} over atoms $y_1,\dots,y_n,x_1,\dots,x_m$. Given φ, we construct a causal logic program P_φ as follows:

$$
\begin{array}{lll}
x_k : x_k & & \text{for each } k \in \{1,\dots,m\} \\
t\ : t & & \\
x_k : \psi_i \leftarrow t & & \text{if } L_{ij} = x_k \text{ for each } i \in \{1,\dots,r\},\ j \in \{1,2,3\} \\
f\ : \psi_i \leftarrow x_k & & \text{if } L_{ij} = \bar{x}_k \text{ for each } i \in \{1,\dots,r\},\ j \in \{1,2,3\} \\
\quad \rho \leftarrow \psi_1,\dots,\ \psi_r & & \\
\quad y_k \leftarrow \mathit{not}\ \bar{y}_k & & \text{for each } k \in \{1,\dots,n\} \\
\quad \bar{y}_k \leftarrow \mathit{not}\ y_k & & \text{for each } k \in \{1,\dots,n\} \\
f\ : \psi_i \leftarrow y_k,\ t & & \text{if } L_{ij} = y_k \text{ for each } i \in \{1,\dots,r\},\ j \in \{1,2,3\} \\
f\ : \psi_i \leftarrow \bar{y}_k,\ t & & \text{if } L_{ij} = \bar{y}_k \text{ for each } i \in \{1,\dots,r\},\ j \in \{1,2,3\} \\
\quad \psi_i \leftarrow \bar{y}_k & & \text{if } L_{ij} = y_k \text{ for each } i \in \{1,\dots,r\},\ j \in \{1,2,3\} \\
\quad \psi_i \leftarrow y_k & & \text{if } L_{ij} = \bar{y}_k \text{ for each } i \in \{1,\dots,r\},\ j \in \{1,2,3\}
\end{array}
$$

Obviously this transformation can be done using logarithmic space. Moreover, it can be shown that φ is true if and only if the c-graph G_{tf} formed by the edge (t,f) is a necessary cause of atom ρ. To wit, first observe that $I(\psi_i) = \sigma_I(L_{i1}) + \sigma_I(L_{i2}) + \sigma_I(L_{i3})$, for any causal stable model I of P_φ, and therefore

$$I(\rho) = \sum \left\{\ \sigma_I(L_{1j_1}) * \ldots * \sigma_I(L_{rj_r})\ \middle|\ j_i \in \{1,2,3\}\ \right\}$$

where

$$
\begin{array}{llll}
\sigma_I(x_k) = t \cdot x_k & \sigma_I(y_k) = t \cdot f \quad \text{if } I \models y_k & \sigma_I(y_k) = 1 \quad \text{if } I \not\models y_k \\
\sigma_I(\bar{x}_k) = x_k \cdot f & \sigma_I(\bar{y}_k) = t \cdot f \quad \text{if } I \not\models y_k & \sigma_I(\bar{y}_k) = 1 \quad \text{if } I \models y_k
\end{array}
$$

The value assigned to atom ρ corresponds to the conjunctive normal form of formula ρ, replacing $\bigwedge$ by $\sum$, $\vee$ by $*$ and L_{ij_i} by $\sigma_I(L_{ij_i})$. Clearly, $\forall x_1,\ldots,x_m\ \rho$ is true if and only if all disjunctions of its conjunctive normal form are valid. The latter is the case for a disjunction if it contains an existential variable y_k assigned to true or two complementary literals of an universal variable x_k. Intuitively, every causal stable model I encodes an assignment to the existential variables $y_1,\ldots,y_n$. If some y_k is assigned to true, then $\sigma_I(y_k) = t \cdot f$. Thus, with every disjunction $L_{1_j1} \vee \ldots \vee y_k \vee \ldots L_{rj_r}$ containing variable y_k, the value $\sigma_I(L_{1_j1}) * \ldots * t \cdot f * \ldots \sigma_I(L_{rj_r})$ is associated, which is obviously smaller than $t \cdot f$. The same also applies to every disjunction containing the literal $\overline{y}_k$ when y_k is assigned to false. Moreover, disjunctions of the form of $L_{1_j1} \vee \ldots \vee x_k \vee \ldots \vee \overline{x}_k \vee \ldots \vee L_{rj_r}$, i.e., containing complementary literals over an universal variable, are assigned a causal term $\sigma_I(L_{1_j1}) * \ldots * t \cdot x_k * \ldots * x_k \cdot f * \ldots * \sigma_I(L_{rj_r})$, which also is smaller than $t \cdot f$. As a consequence, formula φ is true if and only if there exists some causal stable model I (corresponding to an assignment on variables $y_1,\ldots,y_n$) such that $I(p) \leq t \cdot f$ (i.e., formula $\forall x_1,\ldots,x_m\ \rho$ is true under this assignment). The latter is equivalent to deciding whether the causal graph G_{tf} is a brave necessary cause of p (cf. Theorem **??**).

Finally note that, for $n = 0$, i.e., when there are no existentially quantified variables, deciding whether φ is true is coNP-hard and P_φ becomes positive. Therefore:

Theorem 10. *Given a program P, a c-graph G, and an atom p, deciding whether there exists a causal stable model of P such that G is a necessary cause of p is Σ_2^P-complete (coNP-complete when P is positive).* □

5 Related Work and Conclusions

In this work, we have revisited a recent proposal for causal semantics in logic programming [**?**] that assigns a causal explanation to each true atom in a stable model, providing formal definitions for three different causal relationships: being a "sufficient explanation", being a "sufficient cause" and being a "necessary cause". We have shown that, while obtaining the complete causal explanation of an atom has exponential cost, querying whether some cause of p is of any of these three types remains within the first two levels of the polynomial hierarchy (see Figure **??**). Although these results could be reasonable or sometimes even expected, their real significance is that they *affirm the adequacy* of the causal semantics proposed. In fact, this complexity study has led us to disregard a weaker approach previously considered in [**?**] where causal explanations did not guarantee transitivity, since this lack actually yielded higher complexity bounds.

The types of causation defined in the current paper are directly inspired by Hall's classification [**?**]. In that paper, a sufficient explanation is just called "being sufficient" whereas a sufficient cause is said to be "minimally sufficient." There are also other related works on explanations for logic programs as provided by approaches to debugging in ASP [**?**,**?**,**?**,**?**] or other approaches for justifications [**?**,**?**,**?**]. Apart from establishing formal comparisons to these approaches, future work will be focused on three different directions. The next immediate step is implementing the query language to allow queries for positive programs and brave and cautious reasoning for programs with negation. A second line of research is completing the query language and the complexity assessment for *actual causation*. An *actual cause*, in the sense of Mackie [**?**] can be easily defined in this setting as any cause (or causal graph) that is stronger than (i.e., it is a subgraph of) some sufficient cause. This concept can be sometimes convenient since, in order to query if some graph is a sufficient cause, we must provide its complete description, including intermediate events, while for actual causation, we could just check if a partial description of that cause is involved in one of the sufficient explanations. Finally, our most challenging goal is incorporating this type of causal relationships as a new type of literals in program bodies. This would allow representing problems of the form "if sw_3 was a necessary cause for $alarm$ then operator for sw_3 must be penalized" directly as logic program rules.