# Causal Logic Programming*

Pedro Cabalar

Department of Computer Science,
University of Corunna (Spain)
cabalar@udc.es

**Abstract.** In this work, we present a causal extension of logic programming under the stable models semantics where, for a given stable model, we capture the alternative causes of each true atom. The syntax is extended by the simple addition of an optional reference label per each rule in the program. Then, the obtained causes rely on the concept of a *causal proof*: an inverted tree of labels that keeps track of the ordered application of rules that has allowed deriving a given true atom.

## 1 Introduction

Causality is a concept firmly settled in commonsense reasoning. It is present in all kind of human daily scenarios, and has appeared in quite different cultures, both geographically and temporally distant. Paradoxically, although people reveal an innate ability for causal reasoning, the study of causality, or even its very definition, has become a difficult and controversial topic, being tackled under many different perspectives. Philosophers, for instance, have been concerned with the final nature of causal processes and even discussed if there exists so. Logicians have tried to formalise the concept of causal conditionals, trying to overcome counterintuitive effects of material implication. Scientists have informally applied causal reasoning for designing their experiments, but usually disregarded causal information once their formal theories were postulated.

In Artificial Intelligence (AI), we can find two different and almost opposed focusings or currents: (1) *using causal inference*; and (2) *extracting causal knowledge*. Current (1) has been adopted in the area of Reasoning about Actions and Change where most causal approaches have tried to implement some kind of causal derivation in order to solve other reasoning or representational problems. We can cite, for instance, the so-called *causal minimizations* proposed by Lifschitz [1] or by Haugh [2] that were among the first solutions to the well-known Yale Shooting Problem [3], or other later approaches like [4–6] applying causal mechanisms to simultaneously solve the frame and ramification problems. All these formalisms are thought to reason *using* causality but not *about* causality. To put an example, we may use a causal rule like "$A$ *causes* $B$" to deduce that effect $B$ follows from a fact $A$, but we cannot obtain the information "$A$ *has caused* $B$." The only concern is that the rule behaves in a directional way: for instance, we do not want to derive $\neg A$ as an effect of fact $\neg B$. A well-studied formalism based on McCain and Turner's work [5] is the approach of *Non-monotonic Causal Theories* [7] developed by V. Lifschitz and his former doctorate students.

* This research was partially supported by Spanish MEC project TIN2009-14562-C05-04.

Current (2), on the contrary, consists in recognising cause-effect relations like "$A$ *has caused* $B$" from a more elementary, non-causal formalisation[1]. For instance, [10] propose a definition for "*event* $A$ *is an actual cause of event* $B$ *in some context* $C$" in terms of counterfactuals dealing with possible worlds. These possible worlds correspond to configurations of a set of random variables related by so-called *structural equations*. Under this approach, we observe the behaviour of the system variables in different possible situations and try to conclude when "$A$ *has caused* $B$" using the counterfactual interpretation from [11]: had $A$ not happened, $B$ would not have happened. Recently, [12] refined this definition by considering a ranking function to establish the more "normal" possible worlds as default situations. Under this focusing we cannot, however, describe the system behaviour in terms of assertions like "$A$ *causes* $B$" as primitive rules or axioms: this must be concluded from the structural equations.

A third and less explored possibility would consist in treating causality in an *epistemological* way, embodied in the semantics as primitive information, so that we can derive causal facts from it. In this case, the goal is both to describe the scenario in terms of rules like "$A$ *causes* $B$" and derive from them facts like "$A$ *has caused* $B$". This may look trivial for a single and direct cause-effect relation, but may easily become a difficult problem if we take into account indirect effects and joint interaction of different causes. An approach that followed this focusing was [13] that allowed to derive, from a set of causal rules, which sets of action occurrences were responsible for each effect in a given transition system. This approach was limited in many senses. For instance, only actions could form possible causes, but not intermediate events. The causal semantics was exclusively thought for a single transition. Besides, the implementation of causal rules and the inertia default relied on an additional (and independent) use of the nonmonotonic reasoning paradigm of *answer set programming* (ASP) [14, 15], that is, logic programs under the stable model semantics [16].

In this paper we go further and propose to embed causal information *inside* the lower level of ASP. In particular, we are *interested in a formal semantics to capture the causes for each true atom in a given stable model*. To this aim, we extend the syntax by including a label for each rule. Inspired by the *Logic of Proofs* [17], the causes of a given true atom $p$ are then expressed in terms of inverted trees of labels, called *causal proofs*, that reflect the sequence and joint interaction of rule applications that have allowed deriving $p$ as a conclusion. As a result, we obtain a general purpose nonmonotonic formalism that allows both a natural encoding of defaults and, at the same time, the possibility of reasoning about causal proofs, something we may use later to encode high level action languages and extract cause-effect relations among actions and fluents in a more uniform and flexible way.

The rest of the paper is organised as follows. In the next section we explain our motivations and provide a pair of examples. After that, we introduce our semantics for causal proofs, explaining their structure and defining interpretations and valuation of formulas. The next section proceeds to consider positive logic programs explaining how,

[1] Some approaches relying on inductive learning [8, 9] also extract causal information from sets of non-causal observations. However, we do not consider them inside current (2) because the learned theory is of type (1), that is, it allows capturing the domain behaviour but not concluding cause-effect relations like "$A$ *has caused* $B$."

for that case, a concept of model minimality is required. Then, we move to programs with default negation, defining stable models in terms of a straightforward adaptation of the well-known idea of program reduct [16]. Finally, we discuss some related work and conclude the paper.

## 2 Motivation and Examples

Let us see several examples to describe our understanding of a causal explanation for a given conclusion. Causal explanations (or causal proofs) will be provided in terms of rule labels used to keep track of the possible different ways to obtain a derived fact. For readability, we will use different names for labels (usually a single letter) and propositions, but this restriction is not really required. Sometimes, we will also handle unlabelled rules, meaning that we are not really interested in tracing their application for explaining causal effects.

We begin observing that, in order to explain a given derived atom, we will need to handle causes that are due to the *joint interaction* of multiple events. For instance, suppose we have a row boat with two rowers, one at each side of the boat. The boat will only move forward $fwd$ if both rowers strike at a time. We can encode the program as:

$$p : port \quad s : starb \quad port \wedge starb \rightarrow fwd$$

where we labelled the facts $port$ (port rower made a stroke) and $starb$ (starboard rower made a stroke) respectively using $p$ and $s$. Suppose that, in this first example, we are only interested in keeping track of actions (in this case, the labelled facts) and that we leave the rule for $fwd$ unlabelled. From this program we expect concluding not only that $fwd$ (the boat moves forward) is true, but also that its cause is $\{p, s\}$, that is, the *simultaneous* interaction of both strokes.

On the other hand, we will also need considering alternative (though equally effective) causes for the same conclusion. For instance, if we additionally have a following wind, the boat moves forward too:

$$w : fwind \qquad fwind \rightarrow fwd$$

so that we have now two alternative and independent ways of explaining $fwd$: $\{w\}$ and $\{p, s\}$.

From these examples, we conclude that in order to explain a conclusion, we will handle a set of alternative sets of individual events, so that the full explanation for $fwd$ above would be the set $\{\{w\}, \{p, s\}\}$ of its two alternative causes.

Apart from recording labels for facts, we may be interested in a more detailed description that also keeps track of the applied rules. To illustrate the idea, take the following example. Some country has a law $l$ that asserts that driving drunk is punishable with imprisonment. On the other hand, a second law $m$ specifies that resisting arrest has the same effect. The execution $e$ of a sentence establishes that any punishment to imprisonment is made effective unless the accused is exceptionally pardoned. Suppose that some person drove drunk and resisted to be arrested. We can capture this scenario

with the next program:

$$\begin{array}{ll} l : drive \wedge drunk \rightarrow punish & d : drive \\ m : resist \rightarrow punish & k : drunk \\ e : punish \wedge \neg pardon \rightarrow prison & r : resist \end{array}$$

We want to conclude that $punish$ holds because of two alternative causes. The first one is the application of law $l$ on the basis of the joint cause $\{d, k\}$. We will denote this as $l \cdot \{d, k\}$. Similarly, the second one would be due to resistance to arrest $m \cdot \{r\}$. These two causes are independent, so the explanation for $punish$ would contain two alternative causes: $\{l \cdot \{d, k\}\}$ and $\{m \cdot \{r\}\}$. Finally, as there is no evidence of $pardon$ we would conclude that the two independent causes for $prison$ are inherited from $punish$ plus the sentence execution $e$, that is: $\{e \cdot \{l \cdot \{d, k\}\}\}$ and $\{e \cdot \{m \cdot \{r\}\}\}$. We proceed next to formalise these ideas.

## 3 A semantics for causal proofs

A *signature* is a pair $\langle At, Lb \rangle$ of finite sets that respectively represent the set of *atoms* (or *propositions*) and the set of *labels* (or *causal events*). A formula $F$ is defined by the grammar:

$$F ::= p \mid \bot \mid F_1 \wedge F_2 \mid F_1 \vee F_2 \mid l : F_1 \rightarrow F_2$$

where $p \in At$ is an atom and $l$ can be a label $l \in Lb$ or the *null* symbol $\epsilon \notin Lb$. Symbol $\epsilon$ is used when we do not want to label an implication, so that we allow an unlabelled formula $\varphi \rightarrow \psi$ as an abbreviation of $\epsilon : \varphi \rightarrow \psi$. We write $\neg\varphi$ to stand for the implication $\varphi \rightarrow \bot$, and write $\top$ to stand for $\neg\bot$. We also allow labelling non-implicational formulas $l : \varphi$ with some non-null label $l \in Lb$, so that it stands for an abbreviation of $l : \top \rightarrow \varphi$. A *theory* is a finite set of formulas that contains no repeated labels (remember $\epsilon \notin Lb$).

The semantics will rely on the following concept.

**Definition 1 (Causal tree).** *A* causal tree *for a set of labels $Lb$ is a structure $l \cdot C$ where $l \in Lb$ is a label (the* root*) and $C$ is, in its turn, a (possibly empty) set of causal trees.* □

The informal reading of $l \cdot C$ is that "the set of causal trees in $C$ are used to apply $l$." We can graphically represent causal trees as regular trees of labels, $l$ being the root and $C$ the child trees, which cannot contain duplications (remember $C$ is a set). We depict trees upside down (the root in the bottom) and with arrows reversed, as in Figure 1, since this better reflects the actual causal direction, as explained before. In the figure, $t_1$ and $t_2$ are causal trees but $t_3$ is not, since there exists a node (the root $a$) with two identical child trees (the leftmost and rightmost branches). We can deal with usual graph-theoretical terminology for trees so that, for instance, a causal tree with no children $l \cdot \emptyset$ is called a *leaf* (we will just write it $l$ for short). Similarly, the *height* of a causal tree is the length of the longest path to any of its leaves.

Note that, for any non-empty $Lb$, we will have an infinite set of causal trees. To see why, it suffices to observe that just taking $Lb = \{l\}$ we can build, among others, the

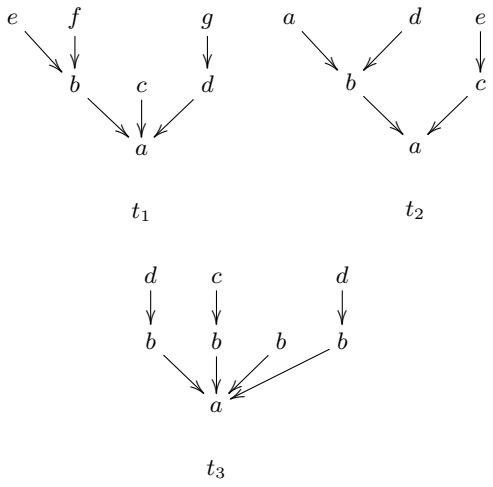



**Fig. 1.** Three examples of (reversed) trees of labels. $t_1$ and $t_2$ are causal trees.

infinite sequence of causal trees $l$, $l \cdot \{l\}$, $l \cdot \{l \cdot \{l\}\}$, etc. However, as we can see, most trees in $\mathcal{T}$ are not very interesting. Many of them contain some subtree $l \cdot C$ where $l$ occurs in $C$ – this means that we are using $l$ to conclude $l$. When this happens, we say that $l \cdot C$ is a *self-supported* causal tree. For instance, $t_2$ in Figure 1 is self-supported. Anything we could explain using $t_2$ can be also explained using leaf $a$ alone. Any causal tree containing a self-supported subtree is said to be *redundant*.

**Definition 2 (Causal proof).** *A* causal proof $l \cdot C$ *is a non-redundant causal tree.* □

We write $\mathcal{P}_{Lb}$ to stand for the set of all possible causal proofs for a set of labels $Lb$. If $Lb$ is finite, $\mathcal{P}_{Lb}$ is also finite and its cardinality is given below.

**Proposition 1.** *The number of causal proofs* $|\mathcal{P}_{Lb}|$ *that can be formed with a set* $Lb$ *of* $n$ *different labels is given by the recursive function:*

$$f(n) = \begin{cases} 1 & \text{if } n = 1 \\ n\ 2^{f(n-1)} & \text{if } n > 1 \end{cases}$$

□

For instance, with the pair of labels $Lb = \{a, b\}$ we get 4 possible causal proofs $\mathcal{P}_{Lb} = \{a, b, a \cdot \{b\}, b \cdot \{a\}\}$.

We define a *cause* $C$ as any (finite) set of causal proofs, that is, $C \in \mathbf{2}^{\mathcal{P}_{Lb}}$. We write $\mathcal{C}_{Lb} = \mathbf{2}^{\mathcal{P}_{Lb}}$ to refer to the set of all possible causes for a set of labels $Lb$. An interesting observation is that given any causal proof $l \cdot C$, the set $C$ forms a cause.

As we explained before, the intuitive meaning of a cause is that it collects a set of causal proofs whose joint interaction suffices to explain a given fact or formula. On the

other hand, we may have a set of causes that are independent alternative explanations. Our semantics will consist, therefore, in assigning a set of causes (that is, a set of sets of causal proofs) to each formula. However, as happened with redundant trees, a set of causes may easily introduce irrelevant information. Consider the following example.

$$\begin{array}{lrl} a : p & p \to r & r \to s \\ & b : p \to q & q \to s \end{array}$$

Fact $s$ can be obtained following two different paths: one following the implications in the first row, leading to cause $\{a\}$; and the other one following the second row of implications and leading to cause $\{b \cdot \{a\}\}$. The causal proof in the first case, $a$, actually forms a subtree of $b \cdot \{a\}$. So, from a causal point of view, the latter is not a fully independent alternative, since with the simple application of $a$ it will always *suffice* to get $s$. Of course, it may be objected that the number of implications for $\{a\}$ is not smaller (it could even be larger). However, all of them are unlabelled, and so, irrelevant for causality steps – we can think of them as "instantaneous." Therefore, we will be interested in dealing with alternative causes that are in some sense minimal (this will be formalised next). Note, however, that inside a cause, we would not want to force minimal causal proofs. For instance, if we modify the previous example as follows:

$$\begin{array}{lrl} a : p & p \to r & q \wedge r \to s \\ & b : p \to q & \end{array}$$

now $s$ requires *both* $q$ and $r$. It seems obvious that $b$ necessarily participates in proving $s$, so we would get the joint cause $\{a, b \cdot \{a\}\}$ for explaining $s$, although one of its proofs is a subproof of the other.

Let us capture now these ideas in a formal way. We mutually define relations of *subproof* and *subcause* denoting them, by abuse of notation, with the same symbol $\preceq$.

**Definition 3 (subproof/subcause).** *Let $C, C'$ denote causes, $l, l'$ labels and $t, t'$ causal proofs. Then:*

- *$l \cdot C$ is a* subproof *of $l' \cdot C'$, written $l \cdot C \preceq l' \cdot C'$, when:*
  - *Either $l = l'$ and $C \preceq C'$;*
  - *or $\{l \cdot C\} \preceq C'$.*
- *$C$ is a* subcause *of $C'$, written $C \preceq C'$, when:*
  *for all $t \in C$ there is some $t' \in C'$ such that $t \preceq t'$.*

□

From a graphical point of view, $t$ is a subproof of $t'$ if it constitutes a subtree, i.e., any tree formed with a subset of nodes and a subset of edges from $t'$. We can also see the subcause relation $C \preceq C'$ by understanding that any proof like $l \cdot C$ will be a subtree of $l \cdot C'$ for any $l$. From this observation, it is quite easy to see that both $\preceq$ are partial order relations. Obviously, $\emptyset \preceq C$ for any cause $C$, so $\emptyset$ is the smallest subcause. Figure 2 shows three subproofs of $t_1$ in Figure 1.

This conclusion is wrong. The subcause relation is, in fact, *weaker* than the subtree relation. In fact, $\preceq$ is just a preorder, i.e., it satisfies reflexivity and transitivity, but not

antisymmetry. As a counterexample with proof trees: $t = a \cdot \{b, b \cdot \{c\}\}$ and $t' = a \cdot \{b\}$, where both $t \preceq t'$ and $t' \preceq t$ but $t' \neq t$.

The reader is referred to [18] for a corrected and substantially improved approach. The correction implies representing both proof trees and causes as a graphs of labels, and defining $C_1 \preceq C_2$ as $C_1 \subseteq C_2^*$ where $C_2^*$ is the reflexive and transitive closure of $C_2$ and $\subseteq$ is the standard subgraph relation. The rest of results in the current paper are subsumed by the new version [18].

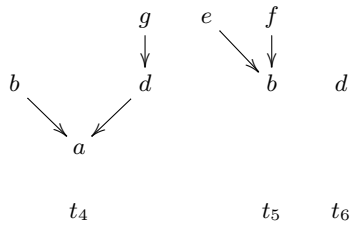



**Fig. 2.** Three examples of subproofs of $t_1$.

Sets of causes $\mathbf{S} \in \mathbf{2}^{\mathcal{C}_{Lb}}$ will be represented with capital boldface letters. By $\min(\mathbf{S})$ we denote the set of $\preceq$-minimal causes from $\mathbf{S}$, formally:

$$\min(\mathbf{S}) \stackrel{\text{def}}{=} \{A \in \mathbf{S} \mid \text{there is no } B \in \mathbf{S} \text{ s.t. } B \preceq A\}$$

We define the application of $l \in (Lb \cup \epsilon)$ to a set of causes $\mathbf{S}$, written $l \odot \mathbf{S}$, as the set of singleton causes $l \odot \mathbf{S} \stackrel{\text{def}}{=} \{\{l \cdot C\} \mid C \in \mathbf{S}\}$. For convenience, the expression $\{\epsilon \cdot C\}$ will be understood as an alternative notation for cause $C$. Using this definition, it is easy to see that $\epsilon \odot \mathbf{S} = \mathbf{S}$.

A *causal value* is a set of causes that are minimal. Formally, the set of causal values $\mathcal{V}_{Lb}$ is defined as:

$$\mathcal{V}_{Lb} \stackrel{\text{def}}{=} \{\mathbf{V} \in \mathbf{2}^{\mathcal{C}_{Lb}} \mid \mathbf{V} = \min(\mathbf{V})\}$$

A *causal interpretation* is a function $I : At \longrightarrow \mathcal{V}_{Lb}$. As $I(p)$ represents the set of alternative causes for $p$ to be *true*, this means that when $I(p) = \emptyset$ (there is no cause for $p$) we understand that $p$ is *false*[2]. It is easy to see that the minimality condition implies, for instance, that if $\emptyset \in \mathbf{V}$ then $\mathbf{V} = \{\emptyset\}$, for any causal value $\mathbf{V}$. When $I(p) = \{\emptyset\}$ we understand that $p$ is true due to the empty cause $\emptyset$. The empty cause will allow deriving conclusions without tracing their proofs with causal labels and, as we see next, it will represent the concept of "maximal truth." Note the difference between $I(p) = \emptyset$ ($p$ false) and $I(p) = \{\emptyset\}$ ($p$ maximally true).

[2] This is because we will later associate $\neg p$ to *default* negation: there is no cause for $p$. If we were interested in representing causes for $p$ being false, this would mean introducing a second kind of negation, usually called *explicit* or *strong* negation.

We define a partial ordering relation '$\sqsubseteq$' on causal values so that for any $\mathbf{V}, \mathbf{V}' \in \mathcal{V}_{Lb}$:

$$\mathbf{V} \sqsubseteq \mathbf{V}' \stackrel{\text{def}}{=} \text{for all } C \in \mathbf{V} \text{ there is some } C' \preceq C \text{ in } \mathbf{V}'$$

The intuitive meaning of $\mathbf{V} \sqsubseteq \mathbf{V}'$ is that $\mathbf{V}'$ makes an atom (or formula) to be *more justified* than with $\mathbf{V}$. For instance, take the particular case $\mathbf{V} \subseteq \mathbf{V}'$ which obviously implies $\mathbf{V} \sqsubseteq \mathbf{V}'$. This means that $\mathbf{V}'$ is offering additional alternative causes to explain a given fact that were not present in $\mathbf{V}$. In the particular case where $\mathbf{V} = \emptyset$, this would mean that $\mathbf{V}'$ makes the fact true (possibly by several causes) whereas $\mathbf{V}$ makes it false. Now, rather than just using a simple inclusion relation, we further specialize it so that each cause in $C \in \mathbf{V}$ is not necessarily present in $\mathbf{V}'$, but there must be some "more justifying cause" $C' \in \mathbf{V}'$ that subsumes $C$, that is $C' \preceq C$. For instance, take $\mathbf{V} = \{\{a \cdot \{b\}\}, \{c, d\}\}$ and $\mathbf{V}' = \{\{b\}, \{d\}, \{e\}\}$. We have $\mathbf{V} \sqsubseteq \mathbf{V}'$ because in $\mathbf{V}'$, we can find subcauses for all elements of $\mathbf{V}$: $\{b\} \preceq \{a \cdot \{b\}\}$ and $\{d\} \preceq \{c, d\}$.

**Proposition 2.** *$\langle \mathcal{V}_{Lb}, \sqsubseteq \rangle$ constitutes a complete lattice with the following least upper bound (lub) '$\sqcup$' and greatest lower bound (glb) '$\sqcap$':*

$$\mathbf{V} \sqcup \mathbf{V}' \stackrel{\text{def}}{=} \min(\mathbf{V} \cup \mathbf{V}')$$
$$\mathbf{V} \sqcap \mathbf{V}' \stackrel{\text{def}}{=} \min\big(\ \{\ (C \cup C') \mid C \in \mathbf{V}, C' \in \mathbf{V}'\}\ \big)$$

*being the top element $lub(\mathcal{V}_{Lb}) = \{\emptyset\}$ and the bottom element $glb(\mathcal{V}_{Lb}) = \emptyset$.* □

**Proposition 3.** *If $\mathbf{V}$ is a causal value then $l \odot \mathbf{V}$ with $l \in (Lb \cup \epsilon)$ is also a causal value.*

**Definition 4 (Valuation of formulas).** *Given a causal interpretation $I$ for a signature $\langle At, Lb \rangle$, we define the valuation of a formula $\varphi$, by abuse of notation also written $I(\varphi)$, following the recursive rules:*

$$I(\bot) \stackrel{\text{def}}{=} \emptyset$$
$$I(\varphi \wedge \psi) \stackrel{\text{def}}{=} I(\varphi) \sqcap I(\psi)$$
$$I(\varphi \vee \psi) \stackrel{\text{def}}{=} I(\varphi) \sqcup I(\psi)$$
$$I(l : \varphi \to \psi) \stackrel{\text{def}}{=} \begin{cases} \{\emptyset\} & \textit{if } l \odot I(\varphi) \sqsubseteq I(\psi) \\ I(\psi) & \textit{otherwise} \end{cases}$$

□

As explained before, falsity $\bot$ is understood as absence of cause, and thus, $I(\bot) = \emptyset$. The causes of a conjunction are formed with the joint interaction of pairs of possible causes of each conjunct. That is, if $C$ is a cause for $\varphi$ and $D$ a cause for $\psi$ then $C \cup D$ together will form a cause for $\varphi \wedge \psi$, provided that $C \cup D$ results minimal among all unions $C' \cup D'$ of that kind. The causes for a disjunction collects (the minimal elements of) the union of causes of both disjuncts. Finally, the most important part is the treatment

of implication, as it must act as a *proof constructor*. As we can see, we have two cases. In the first case, the implication is assigned $\{\emptyset\}$ (simply true) when any cause for the antecedent $C \in I(\varphi)$ forms a cause for the consequent, in principle, $\{l \cdot C\} \in I(\psi)$ where, as we can see, we "stamp" the application of $l$ as a prefix. We say "in principle" because $I(\psi)$ can also be $\sqsubseteq$-greater (more true) just because $I(\psi) = \{\emptyset\}$ for instance. If the condition of the first case fails, then the implication inherits the causal value of the consequent $I(\psi)$.

We say that a causal interpretation $I$ is a *causal model* of some theory $\Gamma$ if for all $\varphi \in \Gamma$, $I(\varphi) = \{\emptyset\}$.

**Observation 1** *If $Lb = \emptyset$ then valuation of formulas collapses to classical propositional logic with truth values $\emptyset$ (false) and $\{\emptyset\}$ (true).*

Let us see some particular cases of implications. For instance, when $l = \epsilon$, we get:

$$I(\varphi \to \psi) \stackrel{\text{def}}{=} \begin{cases} \{\emptyset\} & \text{if } I(\varphi) \sqsubseteq I(\psi) \\ I(\psi) & \text{otherwise} \end{cases}$$

When $\psi = \bot$ the implication becomes $l : \neg\varphi$ and the condition $l \odot I(\varphi) \sqsubseteq I(\bot) = \emptyset$ amounts to $I(\varphi) = \emptyset$ obtaining the valuation:

$$I(l : \neg\varphi) \stackrel{\text{def}}{=} \begin{cases} \{\emptyset\} & \text{if } I(\varphi) = \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

In particular, we can use this to conclude $I(l : \top) = I(l : \neg\bot) = \{\emptyset\}$. Another interesting particular case is $\varphi = \top$, that is, $I(l : \top \to \psi)$ or $I(l : \psi)$ for short. In this case, the set $l \odot I(\varphi)$ becomes $l \odot \{\emptyset\}$ that is $\{\{l \cdot \emptyset\}\} = \{\{l\}\}$. As a result, we obtain:

$$I(l : \psi) \stackrel{\text{def}}{=} \begin{cases} \{\emptyset\} & \text{if } \{\{l\}\} \sqsubseteq I(\psi) \\ I(\psi) & \text{otherwise} \end{cases}$$

A final degenerate case would be $I(\epsilon : \top \to \psi)$ for which $\epsilon \odot \{\emptyset\} = \{\emptyset\}$ and we get the condition:

$$I(\epsilon : \top \to \psi) \stackrel{\text{def}}{=} \begin{cases} \{\emptyset\} & \text{if } \{\emptyset\} \sqsubseteq I(\psi) \\ I(\psi) & \text{otherwise} \end{cases}$$

which trivially collapses into $I(\epsilon : \top \to \psi) = I(\psi)$.

We extend the ordering relation $\sqsubseteq$ to causal interpretations so that given two of them $I, J$, we write $I \sqsubseteq J$ when $I(p) \sqsubseteq J(p)$ for every atom $p$.

## 4 Positive programs and minimal models

Although we will begin focusing on programs without negation, let us first introduce the general syntax of a logic program. As usual, a *literal* is an atom $p$ (*positive literal*) or its negation $\neg p$ (*negative literal*). A *(labelled) logic program* $P$ is a finite set of *rules* of the form:

$$l : B \to H$$

where $B$ is a conjunction of literals (the rule *body*) and $H$ is a disjunction of literals (the rule *head*). The empty conjunction (resp. disjunction) is represented as $\top$ (resp. $\bot$). We write $B^+$ (resp. $B^-$) to represent the conjunction of all positive (resp. negative) literals that occur as conjuncts in $B$. Similarly, $H^+$ (resp. $H^-$) represents the disjunction of positive (resp. negative) literals that occur as disjuncts in $H$. A logic program is *positive* if $H^-$ and $B^-$ are empty for all rules, that is, if it contains no negations. A positive program is further called a *Horn* program if, for all rules, $H$ is an atom. We assume that all the abbreviations seen before are still applicable. Thus, for instance, a rule with empty body $l : \top \to H$ is also written as $l : H$. A rule like $l : p$, with $p$ an atom, is called a *fact*.

Let us see several simple examples. Consider first the program $P_1$ just consisting of fact $a : p$. The expected behaviour is concluding that $p$ holds because of the only cause $\{a\}$. However, as we saw in the previous section, satisfaction of $a : p$ just requires $\{\{a\}\} \sqsubseteq I(p)$. With one label we can only form one causal proof $a$, and thus, only two causes $\{a\}$ and $\emptyset$. Since causal values must collect minimal causes, we get exactly three possible causal values $\emptyset \sqsubseteq \{\{a\}\} \sqsubseteq \{\emptyset\}$. This means that $I(p) = \{\emptyset\}$ would also satisfy $a : p$. In fact, it is quite easy to see that, for a positive program, there always exists a model assigning $\{\emptyset\}$ to all atoms in $At$. It seems obvious that, as happens with standard (non-causal) logic programs, we are interested in a Closed World Assumption, whose reading here would be: "if something is not known to cause a conclusion, it does not cause it." In practice, this means taking $\sqsubseteq$-*minimal* models.

On the other hand, we still want to accept stronger causal values $I(p) = \{\emptyset\}$. This is because other rules could offer more justification for the same atom $p$. Take, for instance, program $P_2$ consisting of $P_1$ plus the (unlabelled) fact $p$. Now, the only model of $a : p$ and $p$ is $I(p) = \{\emptyset\}$. Informally speaking, an unlabelled fact means that $p$ is "trivially true" and this makes any other rule with $p$ in the head to become redundant (it can be just removed).

Consider now the program $P_3$:

$$a : p \qquad b : p \to p$$

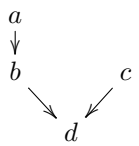
From $a : p$ once again we know that $\{\{a\}\} \sqsubseteq I(p)$. The second rule imposes the restriction $b \odot I(p) \sqsubseteq I(p)$. For instance, $b \odot \{\{a\}\}$ corresponds to causal value $\{\{b \cdot \{a\}\}$ and this is $\sqsubseteq$-smaller than $\{\{a\}\}$ ($a$ is a subtree of $b \cdot \{a\}$). So, it can be seen that a $\sqsubseteq$-minimal model should just make $I(p) = \{\{a\}\}$.

Take now program $P_4$:

$$\begin{array}{ll} a : p & b : p \to q \\ c : r & d : q \wedge r \to s \end{array}$$

One can easily see that a $\sqsubseteq$-minimal model should still make $I(p) = \{\{a\}\}$ as in $P_1$, and a similar reason applies to $I(r) = \{\{c\}\}$. In a next step, we proceed to atoms depending on $p$ and $r$, so from $b : p \to q$ we may conclude $I(q) = \{\{b \cdot \{a\}\}\}$. Finally, in a last step we would get $I(s) = \{\{d \cdot \{b \cdot \{a\}, c\}\}\}$ so that the single causal proof

for $s$ can be graphically depicted as:




This step by step procedure is well-known in standard logic programming. It is usually obtained from the least fixpoint of a *direct consequences* operator [19] for Horn programs. We define an analogous notion for causal Horn programs as follows:

**Definition 5 (Direct consequences).** *We define the* direct consequences *operator $T_P$ for a causal Horn logic program $P$ as a mapping from causal interpretations to causal interpretations such that, for any atom $p$ and interpretation $I$:*

$$T_P(I)(p) \stackrel{\text{def}}{=} \bigsqcup \{\, l \odot I(B) \mid (l : B \to p) \in P \,\} \qquad \square$$

That is, we take the least upper bound of all causal values $l \odot I(B)$ obtained from each pair of label $l$ and rule body $B$ for $p$.

**Proposition 4.** *Operator $T_P$ is monotonic with respect to ordering $\sqsubseteq$ among interpretations.* $\square$

By Knaster and Tarski's theorem [20], Proposition 4 implies that there exists a greatest and a least fixpoint of $T_P$, $gfp(T_P)$ and $lfp(T_P)$ respectively. We can thus extrapolate the classical result from [19] to causal programs:

**Theorem 1.** *A causal Horn program $P$ has a $\sqsubseteq$-least model that coincides with $lfp(T_P)$.* $\square$

We conjecture[3] that $T_P$ is also continuous so it can be computed by iteration on the $\sqsubseteq$-smallest interpretation that makes all atoms false, $I(p) = \emptyset$ for all $p$.

Finally, to illustrate the effect of disjunction, consider the program $P_5$ consisting of the single rule $a : p \vee q$. Some models of this rule satisfy $I(p) = \{\emptyset\}$ or $I(q) = \{\emptyset\}$. For the rest of interpretations, we also have models where $\{\{a\}\} \sqsubseteq I(p)$ or $\{\{a\}\} \sqsubseteq I(q)$. The program has two minimal models $I(p) = \{\{a\}\}, I(q) = \emptyset$ and the dual one $I(p) = \emptyset, I(q) = \{\{a\}\}$.

## 5 Default negation and stable models

Consider now the addition of negation, so that we deal with arbitrary programs. In order to achieve a similar behaviour for default negation to that provided by stable models in the non-causal case, we introduce the following straightforward rephrasing of the traditional program reduct [16].

[3] A formal proof is still under study.

**Definition 6 (Program reduct).** *We define the* reduct *of a program $P$ with respect to an interpretation $I$, written $P^I$, as the result of the following transformations on $P$:*

1. *Removing all rules s.t. $I(B^-) = \emptyset$ or $I(H^-) = \{\emptyset\}$;*
2. *Removing all negative literals from the rest of rules.*

**Definition 7 (Stable model).** *A causal interpretation $I$ is a* stable model *of a causal program $P$ if $I$ is a $\sqsubseteq$-minimal model of $P^I$.*

As an example, take the program $P_6$:

$$a : \neg q \to p \quad b : \neg p \to q \quad c : p \to r$$

As we saw in previous sections, negation $\neg\varphi$ is always two-valued: it returns $\emptyset$ if $\varphi$ has any cause and $\{\emptyset\}$ otherwise. So, when deciding possible reducts, it suffices with considering which atoms, among those negated, will be assigned $\emptyset$. Suppose first we take some $I$ such that $I(p) = \emptyset$, $I(q) \neq \emptyset$. The reduct $P_6^I$ will correspond to:

$$b : \top \to q \quad c : p \to r$$

whose least model is $J(p) = \emptyset, J(q) = \{\{b\}\}, J(r) = \emptyset$. In particular, taking $I = J$ is consistent so we obtain a first stable model. Suppose now we take some $I'$ such that $I'(p) \neq \emptyset$, $I'(q) = \emptyset$. The reduct this time would be:

$$a : \top \to p \quad c : p \to r$$

The least model of this program is $J'(p) = \{\{a\}\}, J'(q) = \emptyset, J'(r) = \{c \cdot \{a\}\}$ which is consistent with the assumption $I' = J'$ so that we get a second stable model. Applying a similar reasoning for the remaining cases, we can easily check that $P_6$ has no more stable models.

## 6 Related Work

Apart from the different AI approaches and orientations for causality we mentioned in the Introduction, from the technical point of view, the current approach can be classified as a *labelled deductive system* [21]. In particular, the work that has had a clearest and most influential relation to the current proposal has been the *Logic of Proofs* [17] (**LP**). We have borrowed from that formalism (most part of) the notation for our causal proofs and rule labellings and the fundamental idea of keeping track of justifications by considering the rule applications. The syntax of **LP** is that of classical logic extended with the construct $t : F$ where $F$ is any formula and $t$ a *proof polynomial*, a term following the grammar:

$$t ::= a \mid x \mid !t \mid t_1 \cdot t_2 \mid t_1 + t_2$$

where $a$ is a *proof constant* (corresponding to our labels) and $x$ a *proof variable*. The meaning of $t : F$ is that $t$ constitutes a proof for $F$. **LP** is an axiomatic system containing the axioms:

**A0**. *Propositional Calculus*
**A1**. $t : F \rightarrow F$ "*reflection*"
**A2**. $t : (F \rightarrow G) \rightarrow (s : F \rightarrow (t \cdot s) : G)$ "*application*"
**A3**. $t : F \rightarrow !t : (t : F)$ "*proof checker*"
**A4**. $s : F \rightarrow (s + t) : F, \;\; t : F \rightarrow (s + t) : F$ "*sum*"

Without entering into further detail, let us overview the main common points and differences between both formalisms. A first important difference comes from the purpose of each approach. While **LP** is thought for capturing a particular logical system, causal logic programs are thought for dealing with non-logical axioms that allow knowledge representation of specific scenarios. Besides, from a technical point of view, **LP** is an axiomatic system, whereas our formalisation relies on a semantic description.

As we can see, proof polynomials are quite similar to our causal proofs. Axiom **A2** looks like a syntactic counterpart of our semantics for labelled implications. However, there also exist some important differences when comparing proof polynomials and causal proofs. For instance, **LP** is much more expressive in the sense that the $t : F$ construction in our approach is exclusively limited to the case in which $t$ is a label. In other words, we have not specified a syntax for expressing that a given cause is assigned to some formula – this information is only obtained as a semantic by-product. As we explain later, the possibility of adding new operators for inspecting causes is left for future study. Another difference is that, while **LP** represents alternative proofs $s$ and $t$ as the polynomial $s + t$, in our causal proofs the ordering or repetition is irrelevant, and so, we simply handle a set of causes. Note also that axiom **A4** does not make sense under our causal reading: if $s$ is a cause for $F$, then not any unrelated $t$ will form a cause $s + t$ for $F$. It is also interesting to observe that the idea of joint causes (that is, the simultaneous interaction of several causal proofs) does not have a syntactic counterpart in **LP**.

Finally, another important difference, especially when thinking about its application to Knowledge Representation, is that **LP** is monotonic whereas our approach allows non-monotonic reasoning and, in fact, is a proper extension of logic programs under the stable model semantics. In this sense, the crucial feature is the introduction of default negation, something that is not present in **LP**.

A preliminary version of the current approach was presented in [22] where most ideas were already present. However, the treatment of causal values and their ordering relation has been considerably improved and generalised now. This has allowed us, for instance, treating disjunction and conjunction respectively as the least-upper and greatest-lower bounds of the causal values lattice. Furthermore, it has also simplified the definition of the direct consequences operator and the proof of its monotonicity.

## 7 Conclusions

We have introduced an extension of logic programming under the stable model semantics that allows dealing with causal explanations for each derived atom $p$ in a stable

model. These explanations are given as sets of alternative, independent causes. In their turn, each cause corresponds to the joint interaction of (one or more) causal proofs, being those used to keep track of the different rule applications that have taken place in the derivation of $p$.

Many open topics remain for future study. For instance, implementation and a complexity assessment are the next immediate steps. We plan to establish a series of formal results relating regular (non-causal) stable model semantics and the information obtained with causal stable models. These results can be exploited for implementation. Regarding expressivity, an interesting topic is the introduction of new syntactic operators for inspecting causal information. Apart from directly representing whether some cause is an explanation for a given formula, we can imagine many different interesting constructs, like checking the influence of a particular event or label in a conclusion, expressing necessary or sufficient causes, or even dealing with counterfactuals. Another interesting topic is removing the syntactic reduct definition in favour of some full logical treatment of default negation, as happens for (non-causal) stable models and their charaterisation in terms of *Equilibrium Logic* [23]. This may allow extending the definition of causal stable models to an arbitrary syntax and to the first order case, where the use of variables in labels may also introduce new interesting features. Finally, as potential applications, our main concern is designing a high level action language on top of causal logic programs with the purpose of modelling some typical scenarios from the literature on causality in AI. Another possible application domain is trying to establish a relation to relevant approaches [24, 25] for debugging in answer set programming.

*Acknowledgements* This work is undoubtedly in debt with Vladimir Lifschitz's pioneering research program and his contributions, among others, to two central areas of Knowledge Representation such as Causal Reasoning and Nonmonotonic Reasoning (NMR). Both my PhD advisor, Ramón P. Otero, and I were first introduced to Vladimir as co-speakers in a seminar on Causality organised by Alessandro Provetti in Milano, a pair of days before the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98). This conference took place in Trento in June 1998, and during its celebration, Vladimir had the courtesy to help us with our formal work relating Otero's Logic of Pertinence to causal reasoning. With his clean and accurate style, he reformulated our results in one piece of paper which put the basis for what constituted later the central part of my PhD dissertation. Today, KR comes back to Italy (KR'12, Rome) and I am still studying Vladimir's papers on NMR and causality much in the same way as fourteen years ago.

## References

1. Lifschitz, V.: Formal theories of action (preliminary report). In: Proc. of the 10th IJCAI, Milan, Italy (1987) 966–972
2. Haugh, B.A.: Simple causal minimizations for temporal persistence and projection. In: Proceedings of the 6th National Conference of Artificial Intelligence. (1987) 218–223
3. Hanks, S., McDermott, D.: Nonmonotonic logic and temporal projection. Artificial Intelligence Journal **33** (1987) 379–413

4. 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 (1995)
5. McCain, N., Turner, H.: Causal theories of action and change. In: Proc. of the AAAI-97. (1997) 460–465
6. Thielscher, M.: Ramification and causality. Artificial Intelligence Journal **1-2**(89) (1997) 317–364
7. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. Artificial Intelligence **153**(1-2) (2004) 49–104
8. Otero, R.P., Varela, M.: Iaction: a system for learning action descriptions for planning. In: Proc. of the 16th Intl. Conf. on Inductive Logic Programming (ILP'06). (2006)
9. Balduccini, M.: Learning action descriptions with A-Prolog: Action language $\mathcal{C}$. In Amir, E., Lifschitz, V., Miller, R., eds.: Proc. of Logical Formalizations of Commonsense Reasoning (Commonsense'07) AAAI Spring Symposium. (2007)
10. Halpern, J.Y., Pearl, J.: Causes and explana- tions: A structural-model approach. part I: Causes. British Journal for Philosophy of Science **56**(4) (2005) 843–887
11. Hume, D.: An enquiry concerning human understanding (1748) Reprinted by Open Court Press, LaSalle, IL, 1958.
12. Halpern, J.Y.: Defaults and normality in causal structures. In: Proc. of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2008). (2008) 198–208
13. Cabalar, P.: A preliminary study on reasoning about causes. In: Proc. of the 6th Intl. Symposium on Logical Formalizations of Commonsense Reasoning. (2003)
14. Marek, V., Truszczyński, M. In: Stable models and an alternative logic programming paradigm. Springer-Verlag (1999) 169–181
15. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence **25** (1999) 241–273
16. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R.A., Bowen, K.A., eds.: Logic Programming: Proc. of the Fifth International Conference and Symposium (Volume 2). MIT Press, Cambridge, MA (1988) 1070–1080
17. Artëmov, S.N.: Explicit provability and constructive semantics. Bulletin of Symbolic Logic **7**(1) (2001) 1–36
18. Cabalar, P., Fandinno, J., Fink, M.: Causal graph justifications of logic programs (2014) unpublished draft available at `http://www.dc.fi.udc.es/~cabalar/cgraphs.pdf`.
19. van Emden, M.H., Kowalski, R.A.: The semantics of predicate logic as a programming language. Journal of the ACM **23** (1976) 733–742
20. Tarski, A.: A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics **5** (1955) 285–309
21. Broda, K., Gabbay, D., Lamb, L., Russo., A.: Compiled Labelled Deductive Systems: A Uniform Presentation of Non-Classical Logics. Research Studies Press (2004)
22. Cabalar, P.: Logic programs and causal proofs. In: Proc. of the 10th Intl. Symposium on Logical Formalization on Commonsense Reasoning (Commonsense'11). AAAI Spring Symposium Series (2011)
23. Pearce, D.: Equilibrium logic. Annals of Mathematics and Artificial Intelligence **47**(1-2) (2006) 3–41
24. 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) 448–453
25. Pontelli, E., Son, T.C., El-Khatib, O.: Justifications for logic programs under answer set semantics. Theory and Practice of Logic Programming **9**(1) (2009) 1–56