Logic Programs and Causal Proofs[*]

Pedro Cabalar

Dept. 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.

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: (1) *using causal inference*; and (2) *extracting causal knowledge*. Orientation (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* (Lifschitz 1987; Haugh 1987) that were among the first solutions to the well-known Yale Shooting Problem (Hanks and McDermott 1987), or other later approaches like (Lin 1995; McCain and Turner 1997; Thielscher 1997) 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 does not manifest the non-directional behaviour of material implication: for instance, we do not want to derive $\neg A$ as an effect of fact $\neg B$.

Orientation (2), on the contrary, consists in recognising cause-effect relations from a more elementary, non-causal formalisation[1]. For instance, (Halpern and Pearl 2005) 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 (Hume 1748): had A not happened, B would not have happened. Recently, (Halpern 2008) 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

[*]This research was partially supported by Spanish MEC project TIN2009-14562-C05-04 and Xunta de Galicia project INCITE08-PXIB105159PR.

[1]Note that some approaches relying on inductive learning (Otero and Varela 2006; Balduccini 2007) also extract causal information from sets of non-causal observations. However, we do not consider them into orientation (2) because the learned theory consists of causal rules for some action language of type (1). In other words, the learned representation allows capturing the domain behaviour but not concluding cause-effect relations like "*A has caused B*."

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. A preliminary approach following this focusing was (Cabalar 2003) that was able 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) (Marek and Truszczyński 1999; Niemelä 1999), that is, logic programs under the stable model semantics (Gelfond and Lifschitz 1988).

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* (Artëmov 2001), 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, for that case, a concept of models 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 (Gelfond and Lifschitz 1988). Finally, we discuss some related work and conclude the paper.

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:

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

where we labelled the facts *port* (port rower made a stroke) and *starb* (starboard rower made a stroke) respectively using p and s. 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 a 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:

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

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.

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 ϵ 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 fundamental concept.

Definition 1 (Causal proof) *A* causal proof *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 proofs.* □

The informal reading of $l \cdot C$ is that "the set of causal proofs in C are used to apply l." We can graphically represent causal proofs as trees of labels, l being the root and C the child trees, which cannot be repeated (remember C is a set). Trees of labels will be depicted 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 proofs 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 tree terminology so that, for instance, a causal proof with no children $l \cdot \emptyset$ is called a *leaf* (we will just write it l for short). Similarly, the *height* of a causal proof is the length of the longest path to any of its leafs.

We define $\mathcal{P}_{Lb}$ as the set of all proofs for a set of labels Lb. When clear from the context, we will remove subindex Lb. Note that $\mathcal{P}$ will be infinite for any non-empty Lb. To see why, it suffices to see that just taking $Lb = \{l\}$ we can build, among others, the infinite sequence of causal proofs l, $l \cdot \{l\}$, $l \cdot \{l \cdot \{l\}\}$, etc. However, as we can see, most causes in $\mathcal{P}$ are not very interesting. Many of them contain some subproof $l \cdot C$ 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 proof. For instance, T_2 in Figure 1 is self-supported. Any causal proof containing a self-supported subproof is said to be *redundant*. For any finite Lb, the set of non-redundant causal proofs is also finite and its cardinal is given below.

Proposition 1 *The number of non-redundant causal proofs that can be formed with a set Lb of n different labels is given by the recursive function:*

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

□

$$\begin{tikzcd}
e \arrow[dr] & f \arrow[d] & & g \arrow[d] \\
& b \arrow[dr] & c \arrow[d] & d \arrow[dl] \\
& & a &
\end{tikzcd}
\qquad
\begin{tikzcd}
a \arrow[dr] & & d \arrow[dl] & e \arrow[d] \\
& b \arrow[dr] & & c \arrow[dl] \\
& & a &
\end{tikzcd}$$

$$T_1 \qquad\qquad T_2$$

$$\begin{tikzcd}
d \arrow[d] & c \arrow[d] & & d \arrow[d] \\
b \arrow[dr] & b \arrow[d] & b \arrow[dl] & b \arrow[dll] \\
& a & &
\end{tikzcd}$$

$$T_3$$

Figure 1: Three examples of (reversed) trees of labels. T_1 and T_2 are causal proofs.

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. As before, subindex Lb is removed when there is no ambiguity. An interesting observation is that given any causal proof $l \cdot C$, the set C forms a cause. *Non-redundant* causes are those that can be formed with non-redundant causal proofs.

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

A *causal interpretation* is a function $I : At \longrightarrow \mathcal{V}$ where $\mathcal{V}$ is the set of *causal values* defined as:

$$\mathcal{V} \ \stackrel{\text{def}}{=}\ \mathbf{2}^{\mathcal{C} \setminus \{\emptyset\}} \cup \{\{\emptyset\}\}$$

That is, a causal value is either a set of non-empty causes, or the singleton $\{\emptyset\}$ exclusively containing the empty cause. Intuitively, $I(p)$ 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]. Note the difference with respect to $I(p) = \{\emptyset\}$ meaning that p is true due to the empty cause $\emptyset$, that

[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.

is, without further justification. 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."

We define an ordering relation '$\sqsubseteq$' on $\mathcal{V}$ causal values so that: (1) for any $S, S' \in \mathcal{V}$ different from $\{\emptyset\}$, $S \sqsubseteq S'$ iff $S \subseteq S'$; and (2) for any $S \in \mathcal{V}$, $S \sqsubseteq \{\emptyset\}$.

Proposition 2 *$\langle \mathcal{V}, \sqsubseteq \rangle$ constitutes a poset with top element $\{\emptyset\}$, bottom element $\emptyset$ and least upper bound $S \sqcup S'$ defined by:*

$$S \sqcup S' \stackrel{\text{def}}{=} \begin{cases} \{\emptyset\} & \text{if } S = \{\emptyset\} \text{ or } S' = \{\emptyset\} \\ S \cup S' & \text{otherwise} \end{cases}$$

Definition 2 (Valuation of formulas) *Given a causal interpretation I for a signature $\langle At, Lb \rangle$, we define the valuation of a formula φ, by abuse of notation also written $I(\varphi)$, following the recursive rules:*

$$\begin{aligned} I(\bot) &\stackrel{\text{def}}{=} \emptyset \\ I(\varphi \wedge \psi) &\stackrel{\text{def}}{=} \{C \cup D \mid C \in I(\varphi) \text{ and } D \in I(\psi)\} \\ I(\varphi \vee \psi) &\stackrel{\text{def}}{=} I(\varphi) \sqcup I(\psi) \\ I(l : \varphi \to \psi) &\stackrel{\text{def}}{=} \begin{cases} \{\emptyset\} & \text{if } l \odot I(\varphi) \subseteq I(\psi) \\ I(\psi) & \text{otherwise} \end{cases} \end{aligned}$$

$\square$

Proposition 3 *For any interpretation I and formula α, $I(\alpha)$ is a causal value.*

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 φ and D a cause for ψ then $C \cup D$ together is a cause for $\varphi \wedge \psi$. The causes for a disjunction is the union of causes of both disjuncts, or $\{\emptyset\}$ if one of them is also $\{\emptyset\}$. 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 $\{l \cdot C\} \in I(\psi)$ where, as we can see, we "stamp" the application of l as a prefix. If this is not the case, 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 Γ 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:

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

When $\psi = \bot$ the implication becomes $l : \neg\varphi$ and the condition $l \odot I(\varphi) \subseteq I(\bot) = \emptyset$ 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:

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

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

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

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

We define an ordering relation $\sqsubseteq$ among causal interpretations so that given two of them I, J, we write $I \sqsubseteq J$ when $I(p) \sqsubseteq J(p)$ for any atom p.

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, that is, if it contains no negations. 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 consisting of the single fact $a : p$. Models of $a : p$ must satisfy either $\{a\} \in I(p)$ or $I(p) = \{\emptyset\}$. If $\{a\} \in I(p)$ we could have, in fact, any *arbitrary* additional cause in $I(p)$ and the interpretation would still keep being a model. Consider now the program P_2:

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

If we choose a model with $\{a\} \in I(p)$, we must have either $\{b \cdot \{a\}\} \in I(q)$ or $I(q) = \{\emptyset\}$. But again, the former does not prevent from including additional arbitrary and perhaps unrelated causes in $I(q)$. Besides, for any positive program, we always have a model where $I(x) = \{\emptyset\}$ for any atom x. 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.

In this way, the least model of P_1 is $I(p) = \{\{a\}\}$ (remember that any causal value is smaller than $\{\emptyset\}$) which fits our intuition: p is true because a. Similarly, the least model of P_2 is $I(p) = \{\{a\}\}$, $I(q) = \{\{b \cdot \{a\}\}\}$ again, as expected.

Suppose now we extend P_2 with the two rules:

$$c : r \qquad d : q \wedge r \rightarrow s$$

Apart from maintaining the previous valuations of p and q, the least model will also assign now $I(r) = \{\{c\}\}$ and $I(s) = \{\{d \cdot \{b \cdot \{a\}, c\}\}\}$ so that the single causal proof for s can be graphically depicted as:

$$\begin{tikzcd}
a \arrow[d] & & \\
b \arrow[dr] & & c \arrow[dl] \\
& d &
\end{tikzcd}$$

To see an example of how the top value $\{\emptyset\}$ works, consider now the extension $P_3 = P_2 \cup \{p\}$. Fact p can also be seen as the rule $\epsilon : \top \rightarrow p$ (as we saw, both formulas are equivalent). Since it must be valuated to $\{\emptyset\}$ (as the rest of rules in the program) the only possible model is $I(p) = \{\emptyset\}$. Note that this still satisfies $a : p$, whereas for rule b, we get the condition $\{b\} \in I(q)$ or $I(q) = \{\emptyset\}$. In other words, program P_3 becomes equivalent to:

$$p \qquad\qquad b : \top \rightarrow q$$

It is easy to see that its least model is $I(p) = \{\emptyset\}$, $I(q) = \{\{b\}\}$.

All the example programs we saw in this section contained no disjunction and had a unique minimal model, that is, a *least* model. We conjecture[3] that, using a similar fixpoint construction as in (non-causal) positive logic programs and their direct consequences operator (van Emden and Kowalski 1976), we will obtain that there always exists such a least model for any positive causal logic program. However, this least model is not necessarily finite, as happened in the examples above. Consider now the program P_4:

$$a : p \qquad\qquad b : p \rightarrow p$$

The causal value of $I(p)$ in the least model will be the infinite set of causes $\{a\}$, $\{b \cdot \{a\}\}$, $\{b \cdot \{b \cdot \{a\}\}\}$, etc, although only the first two are non-redundant[4].

[3] Attempting a proof is left for a future extended version of this document.

[4] In fact, when thinking about algorithms to compute this semantics, it would only probably make sense to consider the set of non-redundant causes which is always finite, as we explained before.

Finally, to illustrate the effect 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 $\{a\} \in I(p)$ or $\{a\} \in I(q)$. The program has two minimal models $I(p) = \{\{a\}\}, I(q) = \emptyset$ and the dual one $I(p) = \emptyset, I(q) = \{\{a\}\}$.

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 as that provided by stable models in the non-causal case, we introduce the following straightforward rephrasing of the traditional program reduct (Gelfond and Lifschitz 1988).

Definition 3 (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 4 (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:

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

As we saw in previous sections, negation $\neg\varphi$ is always two-valued: it returns $\emptyset$ if φ 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:

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

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:

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

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.

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* (Broda et al. 2004). In particular, the work that has had a clearest and most influential relation to the current proposal has been the *Logic of Proofs* (Artëmov 2001) (**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. $s : F \rightarrow (s + t) : F, \quad t : F \rightarrow (s + t) : F$ "*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**.

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 these used to keep track of the different rule applications that have taken place in the derivation of p.

This paper is an introductory work that opens a new field for which many open topics remain for future study. A first 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* (Pearce 2006). 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 this is just a first formal definition, complexity results, implementation issues or potential applications are mostly unexplored yet. Among the latter, we plan to design 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 would be its use for debugging in answer set programming, trying to establish a comparison to relevant works in this topic (Gebser et al. 2008; Pontelli, Son, and El-Khatib 2009).

References

Artëmov, S. N. 2001. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic* 7(1):1–36.

Balduccini, M. 2007. Learning action descriptions with A-Prolog: Action language $\mathcal{C}$. In Amir, E.; Lifschitz, V.; and Miller, R., eds., *Proc. of Logical Formalizations of Commonsense Reasoning (Commonsense'07) AAAI Spring Symposium.*

Broda, K.; Gabbay, D.; Lamb, L.; and Russo., A. 2004. *Compiled Labelled Deductive Systems: A Uniform Presentation of Non-Classical Logics.* Research Studies Press.

Cabalar, P. 2003. A preliminary study on reasoning about causes. In *Proc. of the 6th Intl. Symposium on Logical Formalizations of Commonsense Reasoning.*

Gebser, M.; Pührer, J.; Schaub, T.; and Tompits, H. 2008. Meta-programming technique for debugging answer-set programs. In *Proc. of the 23rd Conf. on Artificial Inteligence (AAAI'08)*, 448–453.

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

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

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

Hanks, S., and McDermott, D. 1987. Nonmonotonic logic and temporal projection. *Artificial Intelligence Journal* 33:379–413.

Haugh, B. A. 1987. Simple causal minimizations for temporal persistence and projection. In *Proceedings of the 6th National Conference of Artificial Intelligence*, 218–223.

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

Lifschitz, V. 1987. Formal theories of action (preliminary report). In *Proc. of the 10th IJCAI*, 966–972.

Lin, F. 1995. 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.

Marek, V., and Truszczyński, M. 1999. *Stable models and an alternative logic programming paradigm*. Springer-Verlag. 169–181.

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

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence* 25:241–273.

Otero, R. P., and Varela, M. 2006. Iaction: a system for learning action descriptions for planning. In *Proc. of the 16th Intl. Conf. on Inductive Logic Programming (ILP'06)*.

Pearce, D. 2006. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence* 47(1-2):3–41.

Pontelli, E.; Son, T. C.; and El-Khatib, O. 2009. Justifications for logic programs under answer set semantics. *Theory and Practice of Logic Programming* 9(1):1–56.

Thielscher, M. 1997. Ramification and causality. *Artificial Intelligence Journal* 1-2(89):317–364.

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