

A Preliminary Study on Reasoning About Causes

Pedro Cabalar

Dept. of Computer Science
University of Corunna, SPAIN
cabalar@dc.fi.udc.es

Abstract

This paper presents some preliminary work on causal reasoning about actions studying the causes of derived formulas in a given transition, in terms of subsets of the performed actions. We present a general top-level semantics that allows deciding which actions have resulted relevant for establishing the truth value of a given formula. After that, we propose a practical implementation that deals with a basic causal rule syntax, translated afterwards into logic programming under answer sets semantics.

Introduction

The appeal to causality has sporadically appeared throughout the evolution of research in actions and change in a more or less explicit way. For instance, McCarthy's seminal paper (McCarthy 1959) already included a definition of *causal assertions*, which consisted in delays of an unspecified number of situations between the condition (or cause) and its resulting effect. Another example is the application of the so-called *causal minimizations* (Lifschitz 1987; Haugh 1987) which were proposed among the first solutions to the well known Yale Shooting Problem (Hanks & McDermott 1987). However, the real interest about causality was raised by the study of the *ramification problem*, i.e., the need to provide a compact description of the indirect effects of actions. The first attempt of encoding indirect effects was representing state constraints as material implications. This led to undesired results due to the interference between the inertia default and the contrapositive nature of classical implication. As a result, several researchers proposed replacing material implications by different kinds of causal conditionals which avoided the contrapositive behavior. This idea, in fact, motivated a prolific trend of causal formalisms, just to cite some of them (McCain & Turner 1995; Lin 1995; Thielscher 1997; Otero 1997; Denecker, Theseider, & van Belleghem 1998; Schwind 1999; Shanahan 1999; Giunchiglia *et al.* 2002), that has continued to the present.

Although all these solutions have successfully solved the interaction between the frame and the ramification problems, there exists, however, a surprising common lack in all of them. Causality is only used as a technical tool, but no

real interest about causal information is shown. To put an example, we may *use* causal rules to deduce, for instance, that after shooting a gun, the turkey results dead, but in most cases we *cannot even represent* that the shot was the cause for that effect. This apparently subtle difference in the kind of information we want to infer may easily become a problem of elaboration tolerance. For instance, if we only performed a shot, it is clear that this must be the cause for the turkey's death. However, what if several shots were simultaneously performed, but only one on a loaded gun? Furthermore, what would happen if the turkey's death is not a direct effect of the shots, but of an arbitrary chaining of indirect effects? Clearly, it should be possible to extract the information of which actions caused which derived effects from the causal rules we use for representing the domain.

In this paper, we propose a top-level semantics for deciding the causes or reasons of each derived fluent formula in terms of subsets of the performed actions. Using a similar device to (Otero 1997), this semantics deals with two components: the standard truth valuation, plus a causal relation used for fixing the causes of each formula. We show how this definition of *cause(s) of a formula* is essential for a compact representation of causal rules, since it allows describing effects that result from the causation of a complex expression, and not just of an action occurrence or a fluent literal. To illustrate the use of this semantics, we propose an implementation using logic programming under answer sets semantics as an underlying nonmonotonic formalism.

The rest of the paper is organized as follows. In the next section, we explain in detail our main goals presenting some motivating examples. After that, we proceed to describe the syntax and semantics for the top-level monotonic formalism. Using some transformations from this formalism, the next section shows how a basic causal action language similar to language \mathcal{C} (Giunchiglia *et al.* 2002) can be translated into a suitable logic program that allows deducing the causes of each fluent effect. Finally, we include a brief discussion and some references to related work.

Some motivating examples

In order to make our goals more precise let us see a simple example.

Example 1 (The lamp circuit)

Consider the circuit in Figure 1, introduced in (Thielscher

1997), where a lamp light bulb is on if and only if two switches are closed. \square

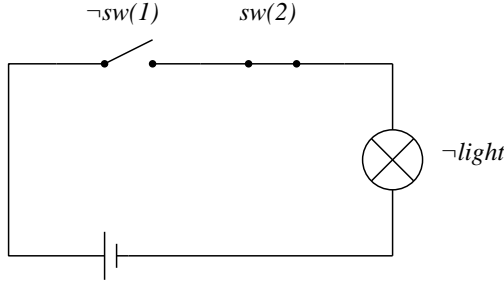


Figure 1: A simple circuit.

Assume that in some successor state we get the configuration represented in the picture. Our purpose is to represent not only the information described by this state (that is, the fluent values), but also how this information has been obtained. That is, which facts have been caused by *which* performed actions, or are just due to inertia. To stress the utility of this, assume that a different person is in charge of toggling each switch, and that a jury must decide who has turned off the light, being legally responsible of some kind of damage. Things can be very different depending on the actions we have performed to achieve this situation. For instance, if we have toggled $sw(2)$ while $sw(1)$ was open, it seems that the light has simply persisted false, and we cannot blame the toggling for this. However, if we moved $sw(1)$ while $sw(2)$ persisted closed, we should obtain that this action is responsible of turning off the light. An interesting case is the one in which both switches were toggled: the light was off and also results off, but disconnecting $sw(1)$ becomes a cause for that, provided that the other person had closed $sw(2)$. This case has an additional interest: it illustrates the fact that we sometimes differentiate a repetition of value due to causation from a real physical persistence.

Let us further consider a pair of new situations. For instance, having both switches connected, if we perform a simultaneous toggling, then any of the two actions seems to be a valid reason for the light being off. This means that we may have several different actions for the same effect. However, if we toggle both switches again, *the two toggling actions together* act as a cause for the light being on. This illustrates that we will need representing each cause a *set* of simultaneously performed actions. Therefore, for each effect, we will deal with a set of causes, being each cause a set of performed actions.

In order to obtain all this information, we should avoid as much as possible the explicit representation of the direct causal influences inside the rule descriptions. For instance, in example 1, we know that the cause for $light$ true will be any cause we can infer for the expression $sw(1) \wedge sw(2)$. So, we expect handling something like:

$$sw(1) \wedge sw(2) \text{ causes } light$$

rather than a multiple set of rules like:

$$\begin{aligned} sw(1) &\text{ causes } light \text{ if } sw(2) \\ sw(2) &\text{ causes } light \text{ if } sw(1) \\ sw(1), sw(2) &\text{ causes } light \end{aligned}$$

which are closer to the description of possible transitions.

Syntax and semantics

The syntax is described starting from a finite set \mathcal{S} of atoms called the *signature* and organized in two disjoint sorts, $\mathcal{S} = \mathcal{A} \cup \mathcal{F}$, where \mathcal{A} and \mathcal{F} respectively contain all the action and the fluent names. Although, we will sometimes use variable arguments for atom names later, we assume that they are actually replaced by all their possible ground instances.

An useful concept for dealing with concurrent actions is the notion of *compound action*, formally defined as any element of $2^{\mathcal{A}}$, that is, any subset of \mathcal{A} . For instance, the compound action $\{toggle(1), toggle(2)\}$ would point out the simultaneous toggling of both switches. As a remark on notation, in the rest of the paper we assume that capital letters A, B, C, \dots will denote compound actions, whereas letters a, b, c, \dots will be used to denote elements of \mathcal{A} , called *elementary* actions as opposed to “compound”. Letters g, h, \dots will stand for fluents whereas p, q, r, \dots will indistinctly denote any atom in the signature. It will also be convenient to represent sets of compound actions, being greek letters α, β, \dots reserved for that purpose.

A formula is recursively defined as follows. Given any atom $p \in \mathcal{S}$, any $A \in 2^{\mathcal{A}}$, and two arbitrary formulas ϕ and ψ , then the following expressions are also formulas:

$$p, \perp, \neg\phi, \phi \wedge \psi, A\phi$$

As we can see, the only non-classical construction is $A\phi$ which can be read as “ A caused ϕ ” and intuitively points out that A is “responsible” of ϕ being true. The rest of classical operators like \vee, \supset, \equiv or \top are defined in terms of \wedge, \neg and \perp in the usual way. Additionally, we will also consider the following (non-classical) derived constructions:

$$\mathbb{C}\phi \stackrel{\text{def}}{=} \bigvee_{A \in 2^{\mathcal{A}}} A\phi \quad (1)$$

$$\mathbb{N}\phi \stackrel{\text{def}}{=} \phi \wedge \neg\mathbb{C}\phi \quad (2)$$

The expression $\mathbb{C}\phi$ (read “caused ϕ ”) just points out that ϕ has been caused true, regardless the action(s) involved in that process. Analogously, $\mathbb{N}\phi$ (read “non-caused ϕ ”) is used to represent that formula ϕ is true without any causal intervention. In fact, when we incorporate later the inertia assumption, $\mathbb{N}\phi$ will also mean that ϕ has *persisted* true.

The *language* \mathcal{L} represents the set of all the syntactically valid formulas (given an implicit signature \mathcal{S}). A formula will be called *classical* if it does not contain causal operators like $A\phi, \mathbb{C}\phi$ or $\mathbb{N}\phi$.

The semantics relies on the basic concept of *interpretation*, defined as a pair $\langle \tau, \rho \rangle$. Component τ is a standard *truth valuation* function $\tau : \mathcal{S} \rightarrow \{\text{t}, \text{f}\}$. We write $\tau_{\mathcal{F}}$ (resp. $\tau_{\mathcal{A}}$) to stand for the submapping of τ exclusively applied on fluents (resp. actions). Clearly, $\tau_{\mathcal{F}}$ corresponds to the usual

| | |
|------------------|------------------|
| ϕ | $\neg\phi$ |
| $\text{f}\alpha$ | $\text{t}\alpha$ |
| $\text{t}\alpha$ | $\text{f}\alpha$ |

| $\phi \wedge \psi$ (where $\alpha \neq \emptyset, \beta \neq \emptyset$) | | | | |
|---|---------------------|-------------------------------|----------------------------------|---------------------|
| $\phi \setminus \psi$ | $\text{f}\emptyset$ | $\text{f}\alpha$ | $\text{t}\alpha$ | $\text{t}\emptyset$ |
| $\text{f}\emptyset$ | $\text{f}\emptyset$ | $\text{f}\emptyset$ | $\text{f}\emptyset$ | $\text{f}\emptyset$ |
| $\text{f}\beta$ | $\text{f}\emptyset$ | $\text{f}(\alpha \cup \beta)$ | $\text{f}\beta$ | $\text{f}\beta$ |
| $\text{t}\beta$ | $\text{f}\emptyset$ | $\text{f}\alpha$ | $\text{t}(\alpha \otimes \beta)$ | $\text{t}\beta$ |
| $\text{t}\emptyset$ | $\text{f}\emptyset$ | $\text{f}\alpha$ | $\text{t}\alpha$ | $\text{t}\emptyset$ |

$$v_I(\perp) = \text{f}\emptyset$$

$$v_I(A\phi) = \begin{cases} \text{t}\{A\} & \text{if } v_I(\phi) = \text{t}\alpha \text{ and } A \in \alpha \\ \text{f}\{\emptyset\} & \text{otherwise} \end{cases}$$

Figure 2: valuation of formulas.

idea of a *fluent state*, whereas τ_A can be handled as a particular compound action, $\{a \in \mathcal{A} | \tau(a) = \text{t}\}$, which contains the currently *performed actions*.

The other component of an interpretation, ρ , is called the *causal relevance relation* and defined as $\rho \subseteq 2^{\mathcal{A}} \times \mathcal{S}$. That is, any element in ρ has the shape (A, p) with the intended meaning: “ A has caused the truth value that τ assigns to p .” A useful alternative way of understanding relation ρ is as a set of boolean mappings $\rho_A : \mathcal{S} \rightarrow \{\text{t}, \text{f}\}$, one for each compound action A , so that $\rho_A(p) = \text{t}$ iff $(A, p) \in \rho$.

Once the idea of interpretation $\langle \tau, \rho \rangle$ is defined, the next usual step is introducing the concept of valuation of any arbitrary formula ϕ . For the truth value, we can just define $\tau(\phi)$ for propositional connectives in the usual way. However, in order to define $\tau(A\phi)$ we will also need to provide a valuation of causal relevance. To this aim, we can extend each function ρ_A in a similar way as we did for τ , that is, applying $\rho_A(\phi)$ for arbitrary formulas. For a more compact representation, another possibility is representing the causal relevance for ϕ , $\rho(\phi)$, as a set of compound actions, so that $A \in \rho(\phi)$ iff $\rho_A(\phi) = \text{t}$.

Definition 1 (valuation function) Given an interpretation $I = \langle \tau, \rho \rangle$ we define its corresponding *valuation function* as the mapping $v_I : \mathcal{L} \rightarrow \{\text{t}, \text{f}\} \times 2^{2^{\mathcal{A}}}$ which assigns to each formula ϕ the pair of values $\tau(\phi)$ $\rho(\phi)$ following the tables in Figure 2. \square

The set operation $\alpha \otimes \beta$ appearing in the table for conjunction is defined as follows:

$$\alpha \otimes \beta \stackrel{\text{def}}{=} \{A \cup B | A \in \alpha, B \in \beta\}$$

As we can see, truth valuation of classical connectives is the standard one. Note also that negation just changes the truth value *without affecting* the causal information. Given this feature, it is easy to see that the table for $\phi \vee \psi$, derived from its definition as $\neg(\neg\phi \wedge \neg\psi)$, would have the same

structure than the table of conjunction changing all¹ the occurrences of t by f and vice versa.

Apart from the tables, we also include a restriction in the valuation of any action atom a :

$$\rho(a) = \begin{cases} \{\{a\}\} & \text{if } \tau(a) = \text{t} \\ \{\emptyset\} & \text{otherwise} \end{cases}$$

We say that an interpretation $I = \langle \tau, \rho \rangle$ satisfies a formula ϕ , written $I \models \phi$, iff $\tau(\phi) = \text{t}$ and that I is *model* of a theory T iff it satisfies all the formulas in T . A *tautology* is a formula ϕ such that $I \models \phi$ for any interpretation I .

The table of conjunction is the most important and complex part of the valuation function. For instance, we distinguished between empty and non-empty (α, β) sets of compound actions. Besides, the reader has probably observed the particular arrangement of values in the ϕ column and the ψ row: we have located non-empty sets α and β in the middle. This arrangement is interesting because allows showing a diagonal symmetry in the table values, from the left-top corner to the right-bottom one. For a better understanding of the table, we will exhaustively study its content using the light bulb scenario as an example.

Consider the formula $sw(1) \wedge sw(2)$ and, as a first case, assume that one of the switches, e.g. $sw(2)$, is off but not caused by any action: $v_I(sw(2)) = \text{f}\emptyset$ (think about the absence of action as a *physical persistence*). Clearly, the conjunction $sw(1) \wedge sw(2)$ must be false regardless any movement in $sw(1)$. Furthermore, we do not need any particular cause to justify that $sw(1) \wedge sw(2)$ is false, i.e., its valuation must be $\text{f}\emptyset$. This observation is important because the possible causes of $sw(1) \wedge sw(2)$ will become later the causes for *light*. So, if we disconnect $sw(1)$, for instance, we would not want to infer that the *light* is caused to be off, knowing that $sw(2)$ was already disconnected! This “short circuit” behavior in persistence can be easily observed on the table: the first row and the first column are fixed to $\text{f}\emptyset$.

Assume now that one switch, e.g. $sw(2)$, persists connected, $v_I(sw(2)) = \text{t}\emptyset$ (the leftmost column in the table). In such a case, it seems clear that the valuation of $sw(1) \wedge sw(2)$ is exclusively established by $sw(1)$ (just think about the circuit as if we would have never had a switch, but a continuous wire instead). This is reflected in the leftmost table column, where we just find a copy of the values for the first conjunct $sw(1)$.

Let us consider now those situations in which no switch has persisted (the four positions in the center square of the table values). These cases result from considering the concurrent execution of both toggling actions. Let us begin with the case $v_I(sw(2)) = \text{f}\alpha$ while $sw(1)$ is connected. Of course, we must get that $sw(1) \wedge sw(2)$ is false (the *light* will be off), but the reason for this is exclusively due to $sw(2)$, i.e., to any cause in α . After all, $sw(1)$ is not “responsible” for the light being off, since it is connected and this would be required for turning the light on. If, instead of being connected, $sw(1)$ had also been caused to be off, but due to another set of causes β , then the falsity of

¹Including the ϕ column and the ψ row.

$sw(1) \wedge sw(2)$ could be indistinctly attributed to any of the switches. In other words, any cause in α or in β is a cause of the conjunction – the valuation must be $\varepsilon(\alpha \cup \beta)$.

Ruling out symmetric situations, the only remaining case is the one in which *both* switches result connected by some causal intervention: $v_I(sw(1)) = \tau\beta$ and $v_I(sw(2)) = \tau\alpha$. For simplicity sake, assume we have only one reason in each case: $\alpha = \{A\}$ and $\beta = \{B\}$ are singletons. The conjunction $sw(1) \wedge sw(2)$ will become true, but the reason for this is the *simultaneous execution* of actions in A and B , i.e., the compound action $A \cup B$. So the resulting valuation would be $\{A \cup B\}$. When α and β contain multiple reasons, we just consider all the combinations $A \cup B$ for any $A \in \alpha$ and any $B \in \beta$.

As for the valuation of the 0-ary connective \perp , we simply fix its truth value as false and consider that it is *never caused*. The reason for this is that \perp is understood as a *constant*, and so, its value does not depend on any causal process. Similarly, constant \top , defined as $\neg\perp$, is always valuated as $\tau\emptyset$. An important consequence of this is that the treatment of tautologies and inconsistencies differs now from classical logic. For instance, formula $p \vee \neg p$ cannot be simply replaced by \top , as we could do in propositional calculus. The reason for this is that, although both expressions are always true, their causal valuation may easily differ: it is enough with having any cause for p .

Finally, let us focus on the valuation of causal expressions like $A\phi$, considering first its truth value. The intended meaning for this formula is that it will be true iff both ϕ is true and A is a cause for ϕ , that is, $A \in \rho(\phi)$. However, in order to give an uniform treatment for this expression with respect to the rest of formulas, we must also provide its causal relevance. To this aim, we propose adopting the criterion of considering it caused by A when true, or caused by the empty compound action \emptyset otherwise². This criterion is also applied to the valuation of action atoms, so that, the only possible causal reason for action a to be true is the compound action $\{a\}$, whereas the reason for being false is the empty compound action.

Another interesting topic about the use of actions is that when we assert a formula like $A\phi$ to express “ A has caused ϕ ” we will probably expect that action A has actually been performed, i.e., $A \subseteq \tau\mathcal{A}$. To achieve this behavior, we include the axiom:

$$A\phi \supset a \quad (3)$$

for any $a \in A$.

Some useful properties that can be easily checked and that will become very useful later are presented next. For instance, the following equivalences are tautologies:

²This criterion has been found technically convenient for the practical representation of causal rules under logic programming. However, we have not found any sound philosophical justification and in fact, we leave open the study of other possibilities. Notice that the decision of causal relevance for causal formulas affects to the interpretation of nested operators $AB\phi$ or to the use of the absence-of-cause as a cause itself, topics that are not covered in this paper.

$$A(\phi \vee \psi) \equiv (A\phi \wedge \neg\mathbb{N}\psi) \vee (A\psi \wedge \neg\mathbb{N}\phi) \quad (4)$$

$$A(\phi \wedge \psi) \equiv (A\phi \wedge \mathbb{N}\psi) \vee (A\psi \wedge \mathbb{N}\phi) \vee \bigvee_{A_1 \cup A_2 = A} (A_1\phi \wedge A_2\psi) \quad (5)$$

$$\mathbb{N}(\phi \vee \psi) \equiv \mathbb{N}\phi \vee \mathbb{N}\psi \quad (6)$$

$$\mathbb{N}(\phi \wedge \psi) \equiv \mathbb{N}\phi \wedge \mathbb{N}\psi \quad (7)$$

Note that \equiv only guarantees that both parts of the equivalence have the same truth valuation, though perhaps their causes may differ. However, these truth-equivalences will be particularly useful when we deal with models of non-nested causal formulas, that is, something like $A\phi$, with ϕ a classical formula. In such a case, the equivalences can be used for decomposing $A\phi$ until causal operators are only applied to literals.

Translating a causal action language into a logic program

The previous framework helps us in deciding the propagation of causes throughout logical formulas. However, it is easy to see that, for its application to action domains, it will not suffice. On the one hand, we will need to cope with formulas at different situations in order to represent the state transitions. This will force us to introduce some kind of non-monotonic reasoning for dealing with the inertia default and avoiding the frame problem. But, on the other hand, non-monotonic reasoning will also be necessary for causes inference itself: for each effect we expect to derive the less causes as possible. Notice that the previous monotonic framework only allows relating the sets of causes with respect to complex formulas, but nothing prevents us to obtain models in which these sets are arbitrarily bigger than expected.

Besides the need of non-monotonic behavior, we must also take into account the ramification problem, trying to obtain a compact rule-based description of indirect effects. As explained in (Lin 1995) or (McCain & Turner 1995), the encoding of rules as material implications may easily lead to a problem of interference between inertia and contraposition. A simple way of avoiding this problem is the use of some kind of non-contrapositive conditional, like the inference rules used in Default Logic³ or (what can be seen as a simpler version) the conditional operator of Logic Programming.

These requirements has led us to consider Logic Programming as a candidate for the (low-level) non-monotonic formalization. Although, probably, other options are equally possible, Logic Programming provides both rule directionality and non-monotonic reasoning (thanks to default negation) in a simple and direct way. Thus, we propose here to follow the methodology established in (Gelfond & Lifschitz 1993) in which, as a first step, a “high level” action language is defined (possibly involving causal expressions) and, afterwards, its semantics can be described in terms of a logic pro-

³An example of work illustrating this use of Default Logic is (Turner 1997).

gram (in that case, interpreted under the *answer sets* semantics). As examples of approaches following this methodology we could cite (Baral, Gelfond, & Proveti 1997; Turner 1997; Lifschitz 1999) or the reference paper (Gelfond & Lifschitz 1998).

The causal rule syntax will be described starting from the signature $\mathcal{S} = \mathcal{A} \cup \mathcal{F}$ and its derived language \mathcal{L} . A *fluent formula* is any classical formula that does not contain action names, and it is further called *fluent literal* if it just consists of a fluent name g or its negation $\neg g$. We define an *action description* as a set of *causal rules* of shape:

$$\phi \text{ causes } \lambda \text{ if } \psi \text{ after } \gamma \quad (8)$$

where ϕ is a classical formula we will call the *causal condition*, λ is a fluent literal called the *effect* and ψ and γ are fluent formulas respectively called *condition* and *precondition*. This syntax has been actually extracted from causal rules of language \mathcal{C} (Giunchiglia *et al.* 2002) with the exception that, in that case, action names may occur in γ instead of ϕ .

A pair of abbreviations will be allowed. For instance, omitting (**if** ψ) or (**after** γ) points out that ψ or γ , respectively, are the formula \top . Besides, the expression:

$$g := \phi \text{ if } \psi \text{ after } \gamma$$

stands for the pair of rules:

$$\begin{aligned} \phi \text{ causes } g \text{ if } \psi \text{ after } \gamma \\ \neg \phi \text{ causes } \neg g \text{ if } \psi \text{ after } \gamma \end{aligned}$$

Intuitively, the expected behavior of a rule like (8) is that once the rule is applicable (we can prove that its condition and precondition are true) we would check whether $A\phi$ is true. If so, the rule should allow deriving $A\lambda$, that is, the effect is caused by the same reason than the causal condition ϕ . Otherwise, the rule is not applied, even when ϕ happens to be true, but not caused.

For practical purposes, we will directly provide the semantics of rules as a translation into a logic program. This is done following several steps. First, we must represent the predecessor and successor fluent states involved in each transition. To do this, for each fluent g , we write g' to represent its value in the predecessor state, leaving the use of g for the successor state. Besides, for any fluent formula γ , we write γ' to stand for the substitution of any fluent name g by g' , i.e., the formula γ valuated at the predecessor state. Second, given a rule r like (8), we define its *primitive body* $pb(r)$ as:

$$pb(r) \stackrel{\text{def}}{=} A\phi \wedge \psi \wedge \gamma'$$

where A actually represents any possible compound action.

Clearly, we can rearrange $pb(r)$ using equivalences (4)-(7) until causal operators are exclusively applied to literals and, in a further step, until we obtain a disjunctive normal form $D_1 \vee \dots \vee D_n$ taking also into account ψ and γ' . Each clause D_i will consist of a conjunction of expressions of any of the shapes $\lambda, \mathbb{N}\lambda, \neg\mathbb{N}\lambda, A\lambda$, where λ is any atom p or its negation $\neg p$. For simplicity sake, we just assume that all these expressions can be directly represented as logic program literals. Then, for each clause D_i , we simply define the logic program rule:

$$A\lambda \leftarrow D_i$$

where A is the union of compound actions A_j , for literals $A_j\lambda$ in D_i .

Together with the translation of rules, some additional axioms need to be incorporated. For instance, in order to provide the right meaning for program literals, we include the rules:

$$\begin{aligned} \mathbb{C}\lambda &\leftarrow A\lambda \\ \lambda &\leftarrow \mathbb{C}\lambda \\ \mathbb{N}\lambda &\leftarrow \lambda, \text{not } \mathbb{C}\lambda \end{aligned}$$

for any fluent literal λ . Inertia can be simply encoded⁴ as:

$$\begin{aligned} g &\leftarrow g', \text{not } \mathbb{C}\neg g \\ \neg g &\leftarrow \neg g', \text{not } \mathbb{C}g \end{aligned}$$

To see how these definitions work, consider the representation of the circuit in example 1. We define $\mathcal{A} = \{t(1), t(2)\}$ and $\mathcal{F} = \{sw(1), sw(2)\}$. The action description consists of the rules:

$$t(N) \text{ causes } sw(N) \text{ after } \neg sw(N) \quad (9)$$

$$t(N) \text{ causes } \neg sw(N) \text{ after } sw(N) \quad (10)$$

$$light := sw(1) \wedge sw(2) \quad (11)$$

where $N \in \{1, 2\}$. First, we replace (11) by the pair of rules:

$$sw(1) \wedge sw(2) \text{ causes } light \quad (12)$$

$$\neg(sw(1) \wedge sw(2)) \text{ causes } \neg light \quad (13)$$

As an example of transformation, $pb(12)$ would be the expression $A(sw(1) \wedge sw(2))$ which, using (5), is transformed into:

$$\begin{aligned} (A sw(1) \wedge \mathbb{N} sw(2)) \vee (A sw(2) \wedge \mathbb{N} sw(1)) \vee \\ \bigvee_{A_1 \cup A_2 = A} (A_1 \phi \wedge A_2 \psi) \end{aligned} \quad (14)$$

This expression is already in disjunctive normal form. From the disjunct, we must construct the set of program rules:

$$A light \leftarrow A sw(1), \mathbb{N} sw(2)$$

varying A in any subset of \mathcal{A} :

$$\emptyset light \leftarrow \emptyset sw(1), \mathbb{N} sw(2) \quad (15)$$

$$\{t(1)\} light \leftarrow \{t(1)\} sw(1), \mathbb{N} sw(2) \quad (16)$$

$$\{t(2)\} light \leftarrow \{t(2)\} sw(1), \mathbb{N} sw(2) \quad (17)$$

$$\{t(1), t(2)\} light \leftarrow \{t(1), t(2)\} sw(1), \mathbb{N} sw(2) \quad (18)$$

Of course, only rule (16) will be really effective, since the rest of causes for $sw(1)$ are impossible in the current domain⁵. The rules for the second disjunct would be symmetrical, whereas the third disjunct leads to the set of rules:

$$A light \leftarrow A_1 sw(1), A_2 sw(2)$$

⁴This representation of inertia has been extracted from (Baral & Lobo 1997) where our $\mathbb{C}\lambda$ is represented there as $ab(\neg\lambda)$.

⁵Adding a reachability study of the dependence graph could rule out the other three useless rules, but we have preferred to maintain the translation of each causal rule independent of the rest of causal rules of the domain.

taking *all* the combinations of compound actions where $A = A_1 \cup A_2$. This leads to the rules:

$$\begin{aligned}
\{t(1)\}light &\leftarrow \{t(1)\}sw(1), \emptyset sw2 \\
\{t(1)\}light &\leftarrow \emptyset sw(1), \{t(1)\}sw2 \\
\{t(1)\}light &\leftarrow \{t(1)\}sw(1), \{t(1)\}sw2 \\
\{t(2)\}light &\leftarrow \{t(2)\}sw(1), \emptyset sw2 \\
\{t(2)\}light &\leftarrow \emptyset sw(1), \{t(2)\}sw2 \\
\{t(2)\}light &\leftarrow \{t(2)\}sw(1), \{t(2)\}sw2 \\
\{t(1), t(2)\}light &\leftarrow \{t(1)\}sw(1), \{t(2)\}sw2 \quad (19) \\
\{t(1), t(2)\}light &\leftarrow \{t(1), t(2)\}sw(1), \emptyset sw2 \\
&\vdots
\end{aligned}$$

but, again, only one rule, (19), is actually effective.

Related work

Although to the best of our knowledge, there is no action approach dealing with the kind of reasoning treated in this paper, there exist many related works which have provided transformations for causal expressions that can be compared to the ones presented here. As examples of these transformations we can cite the work for Event Calculus in (Shanahan 1999) or the approach of *inductive causation* in (Denecker, Theseider, & van Belleghem 1998). Besides, (Thielscher 1997) has also described a way of extracting causal rules from state constraints when we are provided with an additional influence relation that asserts which actions may affect which fluents. A deeper formal comparison is left for future work.

The idea of using an additional influence relation has also been studied in a line of works (Castilho, Gasquet, & Herzig 1999; Castilho, Herzig, & Varzinczak 2002), which allow representing which actions have influenced the value of each fluent. In other words, we can assert whether an atomic action could *participate* or not in the causal process that fixes the value of a given fluent. This, of course, provides less information than the approach we present here. To put an example, when two toggling actions affect the value of *light*, it would be impossible to distinguish whether the cause is their simultaneous occurrence or each one would independently suffice instead. Besides, another important difference is that these approaches require providing the causal dependence information *a priori* (what may easily derive in an elaboration tolerance problem by simply adding new indirect effects), whereas our interest in this paper has been to infer it from the causal rules.

But apart from all these more or less related approaches, the closest one by far is Otero's *Pertinence Logic* (Otero 1997), as we said in the introduction. In fact, in some of the technical aspects, Otero's logic has been used here as a starting point. The purpose of Pertinence Logic is deciding, for each effect, whether it has been caused (regardless the performed actions) or is due to inertia. To this aim, Otero has proposed a logical approach, \mathcal{L}^2 , that deals with two valuation functions: the standard truth-valuation plus an additional one for pointing out the caused (or *pertinent*) formulas. These two valuations allow deciding the truth and perti-

nence value for any complex formula, including causal rules, which are a particular kind of \mathcal{L}^2 conditionals. In Otero's proposal, this framework is combined with a particular minimization policy (similar to the one used in (Lin 1995)) that allows dealing with the typical action scenarios, but simultaneously allows deciding the sets of caused and non-caused facts. Although other action formalizations like predicate *caused* from (Lin 1995) or *occlusion* from (Sandewall 1994) allow handling information of this nature, Otero's approach is characterized by making an exclusive distinction between a caused fact and a fact due to inertia⁶.

Without going into detail, the technical aspects where the current proposal has been inspired by Otero's logic can be easily guessed. For instance, we can see the relevance relation ρ as a generalization of Otero's pertinence valuation where, instead of just pointing out a boolean value (caused/non-caused), we keep track of a set of sets of actions (the particular causes of the formula). Thus, both approaches try to solve the same kind of problem when we restrict ourselves to non-concurrent actions. In that case, if a is the unique performed action at a given transition, then all our caused conclusions would have the shape $\{a\}\phi$, and so, there would not be a real need for distinguishing the set of actions, which is always fixed to $\{a\}$.

Despite of these clear similarities, there also exist some important and (what we consider) fundamental differences. The main one, as expected, comes from the lack of Pertinence Logic for representing the particular causes of each effect. Although this lack is common to most of the action approaches, it becomes crucial, however, with respect to the motivation of this paper. The propagation of causes among complex formulas we present here cannot be trivially guessed from the pertinence valuation. The reason for this is that the latter acts as a simple flag, limited to a distinction between caused effects and inertial ones, losing any possibility of analysis of the relevant causes for each part of the formula. Furthermore, even in the case where we restrict the study to non-concurrent actions, results are significantly different. For instance, the treatment of conjunction (and disjunction) in Otero's work is *independent* of the truth values of the conjuncts. Under $\{\mathcal{L}\}^\epsilon$ definitions, in order to consider a conjunction to be pertinent, it suffices with a conjunct being pertinent, regardless its truth value. Thus, for instance, in Example 1, when we close one switch while the other persists open, the application of pertinence valuation to conjunction would lead to conclude that the light is *caused to be off* while, as we saw before, our proposal here is that the light should persist off in that situation⁷.

⁶Formal comparisons to (Lin 1995) and (McCain & Turner 1997) were included in (Otero & Cabalar 1999). Some examples and an extended discussion on the separation between causation and inertia can be found in (Cabalar 2001).

⁷It must be observed, however, that this difference can be easily avoided by a suitable choice of conjunction and disjunction operators. In fact, truth-sensitive operators for \mathcal{L}^2 can be found in (Cabalar 2001), being defined in terms of \mathbb{C} and \mathbb{N} .

Discussion

The current work has not tried to introduce a whole new action formalization but instead, can be seen as a preliminary proposal for causes deduction which can be probably adapted to other action approaches. Although much work remains to be done, this first attempt should help to open the debate for a new kind of reasoning problems, providing an intelligent agent with the capability of causal introspection. In other words, the agent should not only be able to predict the effects of actions or construct plans depending on its domain representation, but also should be aware of which actions are responsible for each effect. This provides a richer information that can be used to derive new effects. For instance, we could study a scenario where we can obtain the effect of putting Peter in jail, whenever an action performed by him has caused the turkey to be *dead*:

A dead causes jail(peter) if performed(peter, A)

The previous rule does not need to be modified with respect to the possible indirect effects chain between Peter’s actions and the turkey’s death.

Another interesting result from this study is that the use of causal information may help to differentiate between systems which, when just considering fluent state transitions, are equivalent, but have a different causal nature. For instance, consider the example extracted from (Pearl 2000) and shown in Figure 3. Looking at the fluent state transitions, there is no way for distinguishing this circuit from one in which both switches are independently connected in parallel. However, in this circuit the effect of $sw(2)$ is somehow subordinated to the position of $sw(1)$. In our formalism we would use the rule:

$$light := sw(1) \vee sw(2) \quad (20)$$

for representing the two parallel switches and the rule:

$$light := sw(1) \vee \neg sw(1) \wedge sw(2) \quad (21)$$

for the circuit in the figure. Note that, although the (truth) equivalence:

$$sw(1) \vee sw(2) \equiv sw(1) \vee \neg sw(1) \wedge sw(2)$$

is a tautology in propositional logic, when we move to consider the causes of these two formulas, we obtain *different* results. Thus, closing both switches under formalization (20) would cause the light to be on under two independent explanations, $\{toggle(1)\}$ and $\{toggle(2)\}$, whereas the same operation under (21) would only provide the cause $\{toggle(1)\}$ (the second disjunct would become false, and $sw(1)$ alone suffices for explaining the truth of the whole disjunction).

Finally, an interesting open topic is the possible minimization among the set of causes for a given effect. In principle, it could be reasonable to reject causes that are supersets of another causes. However, in some examples, the effects of this further minimization may be not so clear. As an example, consider the circuit presented in Figure 4, forgetting by now the dotted line. If we toggle the three switches simultaneously, we should clearly obtain

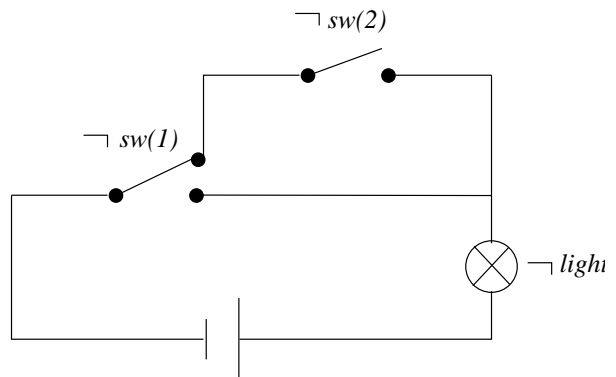


Figure 3: Pearl’s circuit.

two causes for *light*: one due to $toggle(3)$ alone and the other due to $\{toggle(1), toggle(2)\}$ simultaneously. Consider now that switches 1 and 3 are actually connected (with the dotted line) and activated by the same toggling (for instance $toggle(1)$). If we performed $toggle(1)$ and $toggle(2)$ we would of course obtain that $\{toggle(1)\}$ alone is a cause for *light*, since it closes $sw(3)$ now. However, it seems that we should still expect a second cause due to the wire with the pair of closed switches. The only difference with respect to the previous scenario is that, in this case, one of the causes $\{toggle(1), toggle(2)\}$ happens to be a superset of the other one $\{toggle(1)\}$.

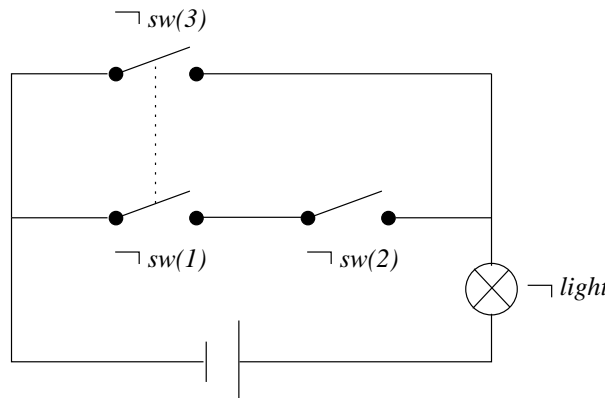


Figure 4: An example of superfluous causes.

For simplicity sake, we have decided not to reject these “superfluous” causes, but we understand that they can be a topic under consideration. For instance, as pointed out by one referee, there seems to exist an interesting relation between rejection of superfluous causes and counterfactual reasoning (Lewis 1973) of the type:

“if switch 2 had not been toggled, the light would still be on.”

that is worth to be studied in the future.

Acknowledgements I want to thank David Lorenzo and Carmen Barro for their advice as independent observers,

pointing out whether the results agreed with their common sense intuitions. This research is partially supported by the Government of Spain, grant TIC2001-0393.

References

- Baral, C., and Lobo, J. 1997. Defeasible specifications in action theories. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 1441–1446.
- Baral, C.; Gelfond, M.; and Proveti, A. 1997. Reasoning about actions: Laws, observations and hypotheses. *Journal of Logic Programming* 31.
- Cabalar, P. 2001. *Pertinence for causal representation of action domains*. Ph.D. Dissertation, Faculdade de Informática, Universidade da Coruña.
- Castilho, M. A.; Gasquet, O.; and Herzog, A. 1999. Formalizing action and change in modal logic I: the frame problem. *Journal of Logic and Computation* 9(5):701–735.
- Castilho, M. A.; Herzog, A.; and Varzinczak, I. J. 2002. It depends on the context! a decidable logic of actions and plans based on a ternary dependence relation. In *Proceedings of the 9th Intl. Workshop on Non-Monotonic Reasoning NMR'02*, 343–348.
- Denecker, M.; Theseider, D.; and van Belleghem, K. 1998. An inductive definition approach to ramifications. *Linköping Electronic Articles in Computer and Information Science* 3(7).
- Gelfond, M., and Lifschitz, V. 1993. Representing action and change by logic programs. *The Journal of Logic Programming* 17:301–321.
- Gelfond, M., and Lifschitz, V. 1998. Action languages. *Linköping Electronic Articles in Computer and Information Science* 3(16).
- Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2002. Nonmonotonic causal theories. (unpublished draft).
- 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.
- Lewis, D. K. 1973. *Counterfactuals*. Harvard University Press, Cambridge, Massachusetts, USA.
- Lifschitz, V. 1987. Formal theories of action (preliminary report). In *Proc. of the 10th IJCAI*, 966–972.
- Lifschitz, V. 1999. Action languages, answer sets and planning. In *The Logic Programming Paradigm: a 25-Year Perspective*. Springer Verlag. 357–373.
- 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.
- McCain, N., and Turner, H. 1995. A causal theory of ramifications and qualifications. In Mellish, C. S., ed., *Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 1978–1984. Montreal, Canada: Morgan Kaufmann.
- McCain, N., and Turner, H. 1997. Causal theories of action and change. In *Proc. of the AAAI-97*, 460–465.
- McCarthy, J. 1959. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, 75–91.
- Otero, R. P., and Cabalar, P. 1999. Pertinence and causality. In *Proc. of the Nonmonotonic Reasoning Actions and Change Workshop (NRAC), at the Intl. Joint Conf. on Artificial Intelligence (IJCAI'99)*, 111–119.
- Otero, R. P. 1997. Pertinence logic for reasoning about actions and change. Technical Report TR-AI-97-01, AI Lab., Dept. of Computer Science, University of A Coruña.
- Pearl, J. 2000. *Causality*. Cambridge University Press.
- Sandewall, E. 1994. *Features and Fluents. A Systematic Approach to the Representation of Knowledge about Dynamical Systems*. Oxford University Press.
- Schwind, C. 1999. Causality in action theories. *Linköping University Electronic Press, Series in Computer and Information Science* 4(4).
- Shanahan, M. P. 1999. The ramification problem in the event calculus. In *Proc. of the Intl. Joint Conf. on Artificial Intelligence (IJCAI'99)*, 140–146.
- Thielscher, M. 1997. Ramification and causality. *Artificial Intelligence Journal* 1-2(89):317–364.
- Turner, H. 1997. Representing actions in logic programs and default theories: A situation calculus approach. *Journal of Logic Programming* 31:245–298.