Causality in Theories of Action

Javier A. Pinto*
Departamento de Ciencia de la Computación, Escuela de Ingeniería,
Pontificia Universidad Católica de Chile, Casilla 306, Santiago, CHILE
jpinto@ing.puc.cl

Abstract

Causation is a concept that plays an essential role in reasoning with *commonsense*. It is a property inherent to the dynamics of changing environments. Therefore, it should follow that causation should also play an essential role in the theories of action and change. However, in this article we postulate that causation is a form of *abstraction*. As a consequence of this view, one should not attempt to model causation as a primitive relationship between entities. Rather, causation should be viewed as an abstract relationship, which can be derived from observations (as a correlation) or derived from our knowledge of the structure of the world.

In this article, we explore the modeling of two types of problems where causation appears to be necessary. We illustrate the problems with some simple examples found in the literature. We show that by decreasing the level of abstraction one can obtain the right results.

1 Introduction.

One of the objectives of the research on theories of action and change is to build languages in which to express our knowledge of the dynamics of the world. On the other hand, commonsense accounts of dynamic processes often refer to *causation* as a relation between world entities. Typical examples of the use of causation are: "Shooting Fred with a loaded gun causes his death," "cigarette smoking may cause cancer," "increase in expenditure causes inflation." The use of causation is pervasive in human discourse. Therefore, it is natural to consider causation as an essential primitive concept, and attempt to incorporate it as a primitive relation in theories of action.

In this article, we postulate that *causation* is an abstraction that hides a complex or unknown reality. For instance, the statement "cigarette smoking may cause cancer," does not mean that the act of smoking sometimes has cancer as a direct consequence. Rather, it states that, by some complex process, that we do not care, or are not prepared to elaborate, cigarette smoking leads to a state in which cancer may develop. On the other hand, it might also mean that there is a statistical correlation between cigarette smoking and cancer. In which case, the statement

*This work has been partly funded by the Fondecyt project number 197-1304.

"cigarette smoking may cause cancer," is taken to mean that it has been observed that cancer develops in a significant number of smokers by some unknown process.

Abstraction is essential in modeling. Particularly so, in modeling the dynamics of the world. The level of abstraction that is chosen has important computational consequences. For instance, suppose that one wants to build a theory about fluids. For this purpose, one might choose to model fluids at the molecular level, and design a language that makes the necessary distinctions to express knowledge about atomic interaction. Representing knowledge at that level would be appropriate, for example, if one wants to model the evaporation of liquids. On the other hand, that level of abstraction might be wrong for the representation of knowledge about filling glasses with tap water, where a much coarser level of granularity seems sufficient. But, if one wants to analyze the molecular properties of fluids, one might not be able to express the laws of fluid behavior, unless the necessary distinctions were incorporated. Therefore, if the level of abstraction is too coarse, one would not get the right results; if the level of abstraction is too detailed, reasoning becomes difficult.

Research in Knowledge Representation has led to the discovery of several problems that are inherent to the enterprise of building theories of action. In this article, we consider two problems. First, the *ramification problem* [1, 5, 17], where one needs to reason about indirect effects of actions. For example, flipping a switch on might turn a light on. However, the indirect effect, like causality, cannot be used in the reverse order. That is, turning the light on cannot have the indirect effect of flipping the switch to the on position. One approach to deal with this problem is to appeal to *causation*, introduced as a primitive relation between entities (for example, see [17, 3, 2]). These entities could be actions/events or fluents (dynamic world properties). For instance, *causes* might be introduced as a predicate between fluents. Then, one would write something like *causes(SwitchOn,LightOn)*, perhaps with certain qualifications, in order to state that whenever the switch is on, the light is also on. If *causes* is treated as asymmetric, then causality can be used to eliminate the possibility that changing the status of the light might change the status of the switch.

Second, we consider the problem of modeling indeterminate actions. For instance, consider the problem of modeling the Russian roulette. In the roulette, a subject takes a revolver with a single bullet, randomly rotates the cylinder of the revolver, and shoots himself. Given that one does not know if the bullet is aligned with the barrel, the effects of shooting are indeterminate. This type of problem has also been addressed with the use of causation [4]. Indeed, one can state that revolving the cylinder causes the bullet to become aligned with the barrel, but it can also cause the bullet not to become aligned with the barrel. Therefore, one would be able to justify the two possible outcomes of randomly rotating the cylinder of the revolver.

In this article we explore the consequences of viewing causation as an abstract notion, for the two types of problems described above. In particular, if one models a particular dynamic reality, and the model fails to give the expected results, the reason might be that the level of abstraction used for the model is too coarse. Therefore, instead of introducing causality to differentiate between *good* and *bad* models of a theory, we postulate the need to change the level of description allowed by the language, and rebuild the theory accordingly. We discuss three examples from the literature. The first two examples (sections 3, 4 and 5) deal with the ramification problem and were taken from [17]. These examples are incarnations of similar examples found in the literature [1, 3]. The example in section 6 is used to illustrate how to handle uncertainty. Before discussing the examples, we outline (section 2) some basic methodological guidelines that we use to model the domains of the examples discussed in the article.

2 How we write theories of action

A common approach to the representation of knowledge about dynamic worlds recognizes actions/events and states to be primitive classes of objects in the world. States can also be associated with metric time. In this article, we work in the Situation Calculus tradition [6]. In particular, we work with an extended Situation Calculus that incorporates several essential notions, like *action occurrence*, *concurrency* and *indeterminacy*. Theories of action are written in the Situation Calculus taking the following notion of a *situation*:

> A situation is a sequence of actions, along with an initial situation (S_0). The state of the world in any given situation is described with a set of fluents (properties), whose truth values do not change within a situation. These fluents describe the state of the world after performing the sequence of actions that lead to the situation, starting in S_0.
>
> The fluents may be called static, because their truth values do not change within a situation. However, fluents may also describe a dynamic situation. For example, a fluent could represent the assertion "the ball is falling due to the gravitational attraction to earth." These assertions can hold throughout the state of a situation.

At the heart of the situation calculus ontology is the notion of action. The world changes from one situation to another as the result of executing actions. In particular, we consider that:

> Actions are instantaneous. Actions with durations can be modeled as a composition of instantaneous actions.
>
> Actions can be initiated by agents or they can be *natural*. Natural actions are *triggered* by nature. Thus, they occur when the state of the world is such that the action must occur.

As discussed later in the article, we need to extend the ontology of actions to allow for concurrency and non-determinism.

Given that the language of the extended Situation Calculus is used to write theories of action, it is natural to ask whether *causation* appears in the Situation Calculus language. From our point of view, a form of causation, that we we call *elementary causation*, is implicit. Elementary causation is:

- A relation between an action and fluents. This elementary causation is specified with *effect axioms*.

- A relation between fluents, implicitly expressed in *ramification state constraints*. The ramification state constraints are *logical* statements that, along with some original set of effect axioms, imply some derived effects. For instance, switching the light off, has as a derived effect to make the room dark. The state constraint would state that the room is dark if and only if the light is off.

- A relation between the state of the world and actions (or events). For example, if the state of the world is such that the pressure inside a container exceeds the pressure capacity of the container, then the container explodes. In this case there is a causal relationship between the *high pressure* condition and an action occurrence. This type of elementary causation is modeled with natural actions, along with the constraint that natural actions occur whenever they are possible.

We regard other forms of causation as abstractions (or compositions) of elementary causation. Thus, one way to interpret the assertion *fluent f causes fluent g to be true* is to state that there is a process that is started when f is true, and this process leads to a state where g is true. In the coming sections we use this view to model some simple examples.

3 The Light and Switches Example

Consider the circuit of the following figure:

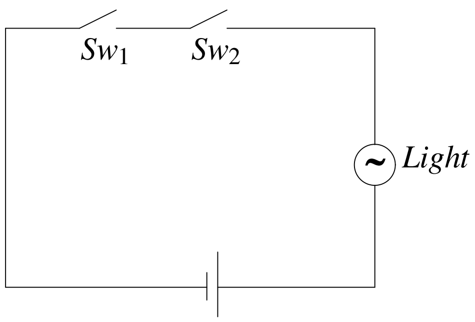

Figure 1: The Light and Switches Example

This example belongs to a class of examples used to describe a form of the ramification problem. The circuit is modeled with a propositional language. The language includes propositional symbols Sw_1, Sw_2 and *Light* (the propositions are true if the corresponding switches are connected, or if the light is on).

The state of the system is modeled with a truth assignment to the propositional letters. Naturally, it seems, some constraints need to be placed on the values of the propositions. In particular, we need to state:

$$Sw_1 \wedge Sw_2 \equiv Light \tag{1}$$

Therefore, any truth assignment to the propositional symbols that is consistent with (1) can describe the state of the circuit. One problem arises when one starts with the state[1]:

$$\{\neg Sw_1, Sw_2, \neg Light\}$$

and Sw_1 is toggled. The only immediate effect of toggling the switch is that Sw_1 becomes true in the new state. However, there is an indirect effect of the toggle action. This effect is that the light comes on. Thus, the resulting state one wants is:

$$\{Sw_1, Sw_2, Light\} \tag{2}$$

Due to the symmetry of the formalization, if one uses a minimal change policy, one would end up with two candidate states, (2) and:

$$\{Sw_1, \neg Sw_2, \neg Light\} \tag{3}$$

Both states have minimal change with respect to the starting state. Clearly, if one understands the behavior of electrical circuits, the state (3) is not wanted. Quite obviously, the problem arises because the model lacks certain essential information. What is missing?

One answer to that question is that we need some way to state that flipping one switch cannot affect the other switch, but that it affects the light. To incorporate this information, one can appeal to *causation*. Thus one can say that Sw_1 **causes** *Light* if Sw_2. This may be taken to mean that if the switch Sw_1 is connected, then it would cause *Light* provided that Sw_2 is also connected. Similarly, (flipping switches) one would have an analogous causal rule for Sw_2 causing *Light* if Sw_1 is on. This causal information, if properly used, would allow us to compute yet another state [17]. After a stable state is reached, the state of the world should be one in which the light is on.

However, there are other ways to deal with this problem. First, observe that the constraint (1) is not really faithful to the reality we are modeling. Rather, it is a simplification. Light bulbs take time to heat up and emit light. Therefore, there will be states of the world where two switches and a light, connected as depicted by figure 1, contradict constraint (1). The key here, is that the passage of time introduces an aspect not accounted for in the propositional model. This is the notion of *state*. After toggling the switch, the circuit changes state to a new state (perhaps unstable). In the new state the light might be off and the two switches might be connected. In these circumstances, the light comes on and a new state is generated.

Therefore, we take the causal relations between the switches and the light to be an abstraction. However, instead of explicitly introducing this abstraction to overcome shortcomings of our models, we can improve the modeling language and decrease the granularity. If a model fails to capture reality, we can attempt to decrease the level of abstraction and be more faithful to the reality we want to capture.

In the next section we propose a different way to model the circuit example, were we show that the type of ramification where causation seemed necessary can be modeled by introducing intervening, perhaps unstable, states.

[1] This means an assignment where *false* is assigned to the propositional symbols Sw_1 and *Light*, and *true* is assigned to Sw_2.

4 A Detailed View of the Light Switches Example

In this section we introduce a new formalization of the light switches examples. We make use of the concurrent Situation Calculus [8, 13], although one could use other specification languages instead (e.g., the Circumscriptive Event Calculus[14]). Concurrency is not necessary to model this example, but it will become useful in the example of next section.

We assume some familiarity with the basics of the Situation Calculus. The concurrent Situation Calculus is a sorted language with sorts $\mathcal{A}, \mathcal{C}, \mathcal{S}, \mathcal{D}$ for simple actions, concurrent actions, situations and domain objects. The sort of the variables used in the examples should be obvious from the context. Furthermore there is a distinguished initial situation denoted with the constant S_0, and a function *do* that, given an action and a situation, yields another situation. Fluents are predicates that take one argument of type situation. Fluents are properties that are static within a situation, but that are dynamic between situations. To model a dynamic domain we describe how the world changes from one situation to another as the result of performing some concurrent action.

The foundational axioms for the concurrent Situation Calculus are[2]:

$$(\forall \varphi).[\varphi(S_0) \wedge (\forall s, c)\ (\varphi(s) \supset \varphi(do(c,s)))] \supset (\forall s)\ \varphi(s), \tag{4}$$

$$do(c_1, s_1) = do(c_2, s_2) \supset c_1 = c_2, \tag{5}$$

$$\neg s < S_0, \tag{6}$$

$$s < do(c, s') \equiv Poss(c, s') \wedge s \leq s', \tag{7}$$

$$Poss(c, s) \equiv (\forall a)[a \in c \supset Poss(a, s)]. \tag{8}$$

Axioms (4) and (5) characterize the space of situations, and axioms (6) and (7) define the relation $<$ between situations. Conceptually, $S < S'$ is true whenever there is a sequence of concurrent actions that, when performed in S lead to S'. Furthermore, all the actions performed must be possible in the situations in which they are performed.

Concurrent actions are treated as sets of primitive actions. Thus, $\in$ is used as a relation between simple actions and concurrent actions. We write $a \in c$ to mean that action a is a part of the concurrent action c. Axiom (8) embodies the assumption that actions can be performed concurrently if and only if all the component actions can be performed individually. In general, this is not correct, since there are actions that can be performed concurrently, but not individually. A more general treatment of concurrency falls outside the scope of this article.

In addition to the axioms (4)-(8) we make use of the notion of *natural action*. Natural actions are not regular *agent actions*. Instead, they are actions that arise due to the nature of the world. One characteristic of these actions is that they must occur whenever they are possible. Thus, we write:

$$Poss(a, s) \wedge natural(a) \supset occurs(a, s). \tag{9}$$

I.e., if a natural action is possible, it must occur. There are several ways in which occurrences can be modeled in the Situation Calculus. One approach is to consider that there is a single branch in

[2] All free variables that appear in formulas are assumed to be universally quantified with maximum scope.

the tree of situations, such that the actions that occur fall in that branch [11, 7]. An alternative is to consider occurrences as constraints that limit the situations that can be considered legal, as in [9, 10]. Here, we use a simplified version of what was proposed in [10]. We define the predicate *legal* for situations, such that:

$$legal(s) \equiv s \geq S_0 \wedge (\forall s', c, a)[s' < s \wedge occurs(a, s') \wedge do(c, s') \leq s \supset a \in c]. \tag{10}$$

Thus, a situation is considered legal if all the actions that lead from S_0 to it are possible (expressed with the $s \geq S_0$ literal), and if all occurrences within the branch $S_0 - s$ are present in the branch. Therefore, to reason with occurrences, we focus on the situations that are *legal*.

To model the circuit of figure 1 we introduce constants Sw_1, Sw_2 and $Light$. We assume that these constants denote distinct individuals. Thus:

$$Sw_1 \neq Sw_2 \wedge Sw_1 \neq Light \wedge Sw_2 \neq Light. \tag{11}$$

Also, we use the fluent *connected* to describe the state of a switch. Thus $connected(Sw, s)$ would be true of a situation where the switch is connected. Similarly, $on(Light, s)$ is meant to state that in a given situation the light is on. We use the fluent $flowing$, which is true of situations where there is current flowing in the circuit. We also add new actions $Start\_flow$ and End_flow. These are the natural actions in this domain:

$$natural(a) \equiv a = Start\_flow \vee a = End\_flow. \tag{12}$$

The basic toggle action is modeled with a function that takes one argument (the switch to be toggled). We need unique name axioms:

$$Start\_flow \neq End\_flow, \tag{13}$$

$$toggle(x) \neq Start\_flow \wedge toggle(x) \neq End\_flow, \tag{14}$$

$$toggle(x) = toggle(y) \supset x = y. \tag{15}$$

Below, we specify the sets of axioms structured according to the style of axiomatization proposed by Reiter [12].

Action precondition axioms

The action precondition axioms are necessary and sufficient conditions for actions to be executable. We have:

$$Poss(toggle(x), s) \equiv x = Sw_1 \vee x = Sw_2, \tag{16}$$

$$Poss(Start\_flow, s) \equiv connected(Sw_1, s) \wedge connected(Sw_2, s) \wedge \neg flowing(s), \tag{17}$$

$$Poss(End\_flow, s) \equiv \neg(connected(Sw_1, s) \wedge connected(Sw_2, s)) \wedge flowing(s). \tag{18}$$

Axiom (16) states that only the two switches can be toggled. Axioms (17) and (18) state the circumstances under which the flow of current may start and may end.

Effect axioms

To describe the effect of toggling a switch, we add the *effect axioms*:

$$Poss(c,s) \wedge toggle(x) \in c \wedge connected(x,s) \supset \neg connected(x, do(c,s)), \tag{19}$$
$$Poss(c,s) \wedge toggle(x) \in c \wedge \neg connected(x,s) \supset connected(x, do(c,s)). \tag{20}$$

The effects of End_flow and $Start\_flow$ are described as:

$$Poss(c,s) \wedge End\_flow \in c \supset \neg flowing(do(c,s)), \tag{21}$$
$$Poss(c,s) \wedge Start\_flow \in c \supset flowing(do(c,s)). \tag{22}$$

State constraints

The state constraint:

$$on(Light, s) \equiv flowing(s). \tag{23}$$

states that the light is on if and only if there is current flowing. This state constraint is a ramification state constraint. Thus, it establishes a truth condition of all states. When an action affects the $flowing$ or the *on* fluents, the action also has, as ramification, an effect on the other fluent.

Following [8], we can derive new effect axioms from the state constraint (23) and the effect axioms (21) and (22):

$$Poss(c,s) \wedge End\_flow \in c \supset \neg on(Light, do(c,s)), \tag{24}$$
$$Poss(c,s) \wedge Start\_flow \in c \supset on(Light, do(c,s)). \tag{25}$$

Successor state axioms

The successor state axioms embody completeness assumptions on the domain description. These axioms are derived from all the effect axioms for the domain. Therefore, for this example, we replace axioms (19)-(25) with:

$$Poss(c,s) \supset on(Light, do(c,s)) \equiv Start\_flow \in c \vee on(Light, s) \wedge End\_flow \notin c, \tag{26}$$
$$Poss(c,s) \supset flowing(do(c,s)) \equiv Start\_flow \in c \vee flowing(s) \wedge End\_flow \notin c, \tag{27}$$
$$\begin{aligned} Poss(c,s) \supset connected(sw, do(c,s)) \equiv\ & toggle(sw) \in c \wedge \neg connected(sw,s) \vee \\ & toggle(sw) \notin c \wedge connected(sw,s). \end{aligned} \tag{28}$$

The initial situation

The only constraints on the possible values assigned to the fluents in the initial situation are the state constraints. Therefore, one should add the state constraints (particularized to S_0) to the theory. We omit this step, and assume that in the initial situation, S_0, the state is one where the second switch is connected, the first switch is disconnected and the light and current flow are off. Thus:

$$\neg connected(Sw_1, S_0) \wedge connected(Sw_2, S_0) \wedge \neg flowing(S_0) \wedge \neg on(Light, S_0). \tag{29}$$

Now, we have the following proposition:

Proposition 4.1 *Let Σ denote the theory with axioms (4)-(18) and (26)-(29). Then:*

$$\begin{aligned} \Sigma \models (\forall c_1, c_2, s).& \\ & legal(s) \wedge s = do(c_2, do(c_1, S_0)) \wedge \\ & toggle(Sw_1) \in c_1 \wedge toggle(Sw_2) \notin c_1 \supset \\ & \qquad \qquad Start\_flow \in c_2 \wedge on(Light, s). \end{aligned} \tag{30}$$

This proposition states that, after toggling Sw_1 in S_0, the current starts flowing and the light comes on[3].

Conclusions for this example.

In this example we show that we are able to obtain the results that we expect from the model. That is, the *Light* is on after the circuit becomes stable. To obtain this result, we explicitly incorporated the dynamic nature of the behavior of the circuit using a sequence of states. Thus, unless we explicitly toggle Sw_2, toggling Sw_1 leads to a state where the *Light* is on. We are able to obtain much more than that. For instance, we can derive explanations, if we observe that the *Light* is on some time after S_0, then there is an intermediate situation where Sw_1 is toggled.

5 Extending the light and switches example

Figure 2 shows a more complex example involving three switches and a relay. The idea here is that the light comes on whenever the switches Sw_1 and Sw_2 are connected. The relay acts to disconnect Sw_2. The relay is active if Sw_1 and Sw_3 are connected.

Let us assume that the initial state of the circuit is one in which Sw_1 is open, Sw_2 and Sw_3 are closed, and the *Light* is off. Then, what happens if we flip Sw_1?

Let us assume that the initial state is such that Sw_1 is disconnected and the other two switches are connected. What happens if the switch Sw_1 is closed? After the switch is closed, and assuming that the other switches were not acted upon, two flows are started. One current goes through the light and the other through the relay. In that situation, the relay opens, and the flow of current through the light ends.

To model this circuit, we use constants Sw_1, Sw_2, Sw_3, *Light*, *Relay*. The fluent $flowing$ now takes one domain argument, in addition to the situation argument. Thus, $flowing(Light, s)$

[3]Proof are omitted for lack of space.

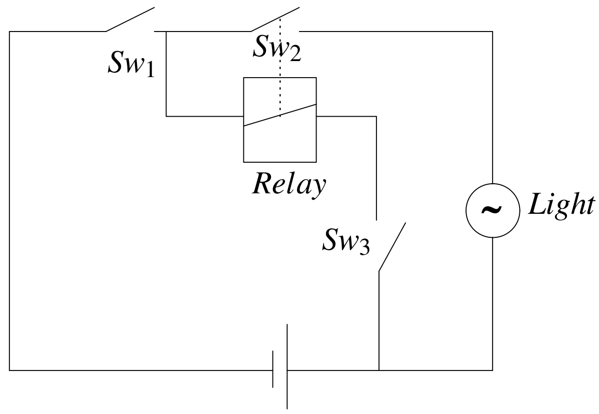

Figure 2: Extended Light and Switches Example

will be true in situations where there is current flowing through the light. We also use the action function symbols $start\_flow$ and end_flow which take a single domain argument.

The action $toggle$ is modeled as before. We incorporate a new action $Open$ which occurs when the relay opens switch Sw_2. We include $Open$ among the natural actions, along with the $start\_flow$ and end_flow actions:

$$\begin{aligned} natural(a) \equiv\ & (\exists x)(a = start\_flow(x) \lor a = end\_flow(x)) \lor \\ & a = Open. \end{aligned} \tag{31}$$

We assume unique name axioms for domain and action terms (we omit them this time).

The structure of the axiomatization is the same as before. Therefore, we have:

Action precondition axioms

$$Poss(toggle(x), s) \equiv x = Sw_1 \lor x = Sw_2 \lor x = Sw_3, \tag{32}$$

$$\begin{aligned} Poss(start\_flow(x), s) \equiv\ & \neg flowing(x, s) \land \\ & [x = Relay \land connected(Sw_1, s) \land connected(Sw_3, s) \lor \\ & x = Light \land connected(Sw_1, s) \land connected(Sw_2, s)], \end{aligned} \tag{33}$$

$$\begin{aligned} Poss(end\_flow(x), s) \equiv\ & flowing(x, s) \land \\ & [x = Relay \land \neg(connected(Sw_1, s) \land connected(Sw_3, s)) \lor \\ & x = Light \land \neg(connected(Sw_1, s) \land connected(Sw_2, s))], \end{aligned} \tag{34}$$

$$Poss(Open, s) \equiv on(Relay, s) \land connected(Sw_2, s). \tag{35}$$

Effect axioms

The effect axioms for $start\_flow$ and end_flow are:

$$Poss(c,s) \wedge start\_flow(x) \in c \supset flowing(x, do(c,s)), \tag{36}$$
$$Poss(c,s) \wedge end\_flow(x) \in c \supset \neg flowing(x, do(c,s)). \tag{37}$$

The effect axioms for $toggle$ are as before:

$$Poss(c,s) \wedge toggle(x) \in c \wedge connected(x,s) \supset \neg connected(x, do(c,s)), \tag{38}$$
$$Poss(c,s) \wedge toggle(x) \in c \wedge \neg connected(x,s) \supset connected(x, do(c,s)). \tag{39}$$

The effect axiom for $Open$ is:

$$Poss(c,s) \wedge Open \in c \wedge connected(Sw_2,s) \supset \neg connected(Sw_2, do(c,s)). \tag{40}$$

State constraints

The state constraint (23) takes the form:

$$on(x,s) \equiv flowing(x,s). \tag{41}$$

From where we can obtain the following effect axioms:

$$Poss(c,s) \wedge end\_flow(x) \in c \supset \neg on(x, do(c,s)), \tag{42}$$
$$Poss(c,s) \wedge start\_flow(x) \in c \supset on(x, do(c,s)). \tag{43}$$

Successor state axioms

From the effect axioms we derive the following successor state axioms:

$$Poss(c,s) \supset on(x, do(c,s)) \equiv start\_flow(x) \in c \vee on(x,s) \wedge end\_flow(x) \notin c, \tag{44}$$
$$Poss(c,s) \supset flowing(x, do(c,s)) \equiv start\_flow(x) \in c \vee flowing(x,s) \wedge end\_flow(x) \notin c, \tag{45}$$
$$\begin{aligned} Poss(c,s) \supset connected(sw, do(c,s)) \equiv\ & toggle(sw) \in c \wedge \neg connected(sw,s) \vee \\ & toggle(sw) \notin c \wedge Open \notin c \wedge connected(sw,s). \end{aligned} \tag{46}$$

Initial situation

Let the initial situation be such that:

$$\begin{aligned} &\neg connected(Sw_1, S_0) \wedge connected(Sw_2, S_0) \wedge connected(Sw_3, S_0) \wedge \\ &\neg flowing(Relay, S_0) \wedge \neg flowing(Light, S_0) \wedge \neg on(Light, S_0) \wedge \neg on(Relay, S_0). \end{aligned} \tag{47}$$

Proposition 5.1 *Let Σ denote the theory with axioms (4)-(10), (31)-(35) and (44)-(47), along with an appropriate set of unique names axioms. Then:*

$$
\begin{aligned}
\Sigma \models (\forall c_1, c_2, c_3, c_4, x, s).& \\
& legal(s) \wedge s = do([c_1, c_2, c_3, c_4], S_0) \wedge \\
& toggle(Sw_1) \in c_1 \wedge toggle(Sw_2) \notin c_1 \wedge toggle(Sw_3) \notin c_1 \wedge \\
& toggle(x) \notin c_2 \wedge toggle(x) \notin c_3 \wedge toggle(x) \notin c_4 \supset \\
& \qquad\qquad start\_flow(Light) \in c_2 \wedge start\_flow(Relay) \in c_2 \\
& \qquad\qquad Open \in c_3 \wedge end\_flow(Light) \in c_4 \wedge \neg on(Light, s).
\end{aligned}
\tag{48}
$$

Therefore, toggling the first switch (and not touching the others) provokes a chain reaction that ends with the light off.

Conclusions for this example

Once again, a shift in the level of abstraction allows us to obtain the expected results. The circuit modeled incorporates a greater level of complexity, which is reflected in the axiomatization. There is a greater number of *unstable* intermediate states, each leading to another state, through natural actions.

It might be argued that this model fails to capture the real behavior of the circuit. Indeed, studying the consequences of proposition 5.1 it follows that the light is on in an intervening state. However, the physical behavior of electrical devices is such that they take time to react. If current starts flowing through the lamp, the light will come on after some ε_l (a few milliseconds, perhaps). Likewise, if current starts flowing through the relay, the relay will react after some ε_r. Depending on the values of these constants, the light will come on only if the relay is slower to react than the light. More likely, one would not know, beforehand, which reaction time is quicker. Therefore, only an experiment would tell us whether the light comes on or not. Also, given the nature of these devices, it may happen that the light comes on sometimes, and does not come on some other times. This happens, for instance, if the behavior of the devices depends on the ambient temperature or some other physical parameter.

Therefore, the results of proposition 5.1 can be considered unsatisfactory, on the grounds that we fail to yield one perfectly reasonable candidate solution. That is, a solution where the light does not come on in any intervening state. To obtain such a result, we need to shift to a lower level of abstraction. We don't have space to discuss the details of such a shift. However, suffice it to say that we need to introduce metric time (see [13, 8] for a discussion on how this can be done). Furthermore, we need to state that relays and lights take time to react. Then, depending on the values of numeric constants, we would obtain the wanted behaviors. However, these behaviors would be obtained because the model is faithful to the reality we want to model, not because we *code in* the solution into the axiomatization.

6 Indeterminacy

Indetermination regarding actions and their effects can be understood in different ways. For instance, consider the action of tossing a coin, which can be considered to be:

> A well known and inherently indeterminate action. Therefore, the situation resulting from performing the action is known, but its state is not.
>
> An indeterminate choice between deterministic actions. The coin can be tossed so that it falls head up or can be tossed so that it falls tail up. The problem is that our ability (or lack of it) limits our choice of action to perform. Thus, we may choose to toss the coin, but we may not be able to choose whether to toss it one way or the other.

We take the second view, that is, we assume that there are no primitively indeterminate actions. Indeterminacy arises because we may not know which action (among a set) is performed. This view has important practical consequences. In particular, the specification of the effects of primitive actions need not account for non-determinacy. Therefore, the effects of primitive actions are specified as before. Indeterminacy is modeled as indeterminate choice between two actions (corresponding to the two ways to toss the coin).

In what follows we model indeterminate actions with a new sort. Each indeterminate action would be a set of concurrent actions.

Given an indeterminate action i, what do we want $do(i,s)$ to denote? There are two answers to this question: First, it is equal to the situation that results from performing one of the actions in i. Second, it is a set of situations, each resulting from performing some action that belongs to i in the situation s. We pursue the second alternative, as described below.

We extend the language of the concurrent Situation Calculus in one essential way. We add two sorts: $\mathcal{I}$ for indeterminate actions, and $\mathcal{S}^{\mathcal{S}}$ for sets of situations. The indeterminate actions are taken to be sets of concurrent actions. In the same manner as concurrent actions are sets of primitive actions. Thus, the indeterminate action $\{\{toss(Head)\},\{toss(Tail)\}\}$ might denote the indeterminate action of tossing a coin. The result from performing an indeterminate action is a set of situations. Each situation in this set is a possible outcome of the indeterminate action. For example, the result of performing the indeterminate action $\{c_1,c_2\}$ in a situation s is the set of situations $\{do(c_1,s),do(c_2,s)\}$. The state of each situation in the resulting set is given by the successor state axioms (as before).

In order to formalize this idea, we overload the symbols in the original vocabulary. The function do takes, in the concurrent Situation Calculus, a concurrent action and a situation as arguments. Now, we define do as a function from $\mathcal{C}\times\mathcal{S}\rightarrow\mathcal{S}$, from $\mathcal{C}\times\mathcal{S}^{\mathcal{S}}\rightarrow\mathcal{S}^{\mathcal{S}}$, from $\mathcal{I}\times\mathcal{S}\rightarrow\mathcal{S}^{\mathcal{S}}$, and from $\mathcal{I}\times\mathcal{S}^{\mathcal{S}}\rightarrow\mathcal{S}^{\mathcal{S}}$. The denotation of do for concurrent actions and situations is as before. The axioms to describe the extended do function are:

$$c \in i \equiv do(c,s) \in do(i,s), \tag{49}$$

$$s \in s^s \equiv do(c,s) \in do(c,s^s), \tag{50}$$

$$c \in i \wedge s \in s^s \equiv do(c,s) \in do(i,s^s). \tag{51}$$

The *Poss* predicate, need also be extended:

$$Poss(c, s^s) \equiv (\forall s).s \in s^s \supset Poss(c, s), \tag{52}$$

$$Poss(i, s^s) \equiv (\forall c).c \in i \supset Poss(c, s^s). \tag{53}$$

We also allow fluent predicates to take a set of situations as arguments. We need the following schema. For every fluent predicate f:

$$(\forall \underline{x}, s^s).f(\underline{x}, s^s) \equiv (\forall s).s \in s^s \supset f(\underline{x}, s). \tag{54}$$

This treatment of fluent truth in a set of situations is reminiscent to the treatment of truth in possible worlds. Thus, we say that f is true in a set if it is necessarily true.

Example

The standard examples for indeterminate actions are coin flippings, coins on checkered boards, Russian roulettes, etc. Let us deal with the coin on a checkered board. If one drops a coin on a checkered board the result will be that either the coin is on a white square, on a black square, or is partially over both. So, let us assume that we have the fluents $onHand, onWhite, onBlack$. The simple actions in this domain are drop actions $drop(x)$ where x is either $White$, $Black$, or $Both$; in addition, we allow a *pickUp* action. The indeterminate action *Drop* can be defined as:

$$Drop = \{\{drop(White)\}, \{drop(Black)\}, \{drop(Both)\}\}$$

Actions and their effects are modeled in the same manner as before. I.e., without considering indetermination. It suffices to explicitly write the successor state axioms for the fluents of this domain:

$$Poss(c, s) \supset onHand(do(c, s)) \equiv onHand(s) \wedge (\forall x).drop(x) \notin c \vee pickUp \in c. \tag{55}$$

$$Poss(c, s) \supset onWhite(do(c, s)) \equiv onWhite(s) \wedge pickUp \notin c \vee drop(White) \in c \vee drop(Both) \in c. \tag{56}$$

$$Poss(c, s) \supset onBlack(do(c, s)) \equiv onBlack(s) \wedge pickUp \notin c \vee drop(Black) \in c \vee drop(Both) \in c. \tag{57}$$

In order to complete the example we need to add state constraints (the coin cannot be on the board and in the hand at the same time), action preconditions (to drop the coin, it has to be in the hand), etc. For lack of space, we do not pursue the formalization of the example further. However, the approach we take to deal with the indetermination should be clear. Performing an indeterminate action results in a set of situations. In the example of the coin on a checkered board, we can infer that after dropping the coin we end up in a set of situations. An interesting research venue is to integrate indeterminacy with Levesque and Lespérance's notion of *knowledge producing actions*, where we could perform an action that would reduce the level of indetermination. Thus, the result of tossing a coin, and looking at the outcome, should yield a single possible situation.

7 Conclusion

In this article we have proposed to treat causality as an abstraction defined in terms of primitive causation. The level of granularity of the axiomatization is defined by primitive causation, expressed with direct action effects, state constraints, and natural occurrences. Choosing a wrong level of granularity will produce flawed reasoning, as exemplified by examples in the AI literature. In future research, we would like to explore the possibility of defining causation, perhaps meta-theoretically, to account for the different views of causation that can be found in the literature (see [15, 16] for a brief review).

How can causation be integrated into a theory of action. If we consider causation to be an abstraction, useful in reasoning processes, we can take the same point of view in building theories of action. For instance, suppose that we can prove that after a fluent f becomes true, a fluent f' becomes necessarily true some time later. Then, we can cache that fact in our knowledge base and avoid expensive reasoning.

Unfortunately, it seems difficult or impossible to make formal statements about causality. The reason is that causality is an informal notion, whose meaning has eluded formalization. This paper proposes a methodological approach to deal with causality. If one is tempted to write down a causal rule to fix a theory, one should always consider the alternative of dissecting causation and eliminate it in favor of more primitive rules that one believes are faithful to the reality being modeled.

References

[1] GINSBERG, M. L., AND SMITH, D. E. Reasoning about Actions I: A Possible Worlds Approach. *Artificial Intelligence 35*, 2 (1988), 165–195.

[2] KARTHA, G. N., AND LIFSCHITZ, V. Actions with Indirect Effects (Preliminary Report). In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR'94)* (1994).

[3] LIN, F. Embracing Causality in Specifying the Indirect Effects of Actions. In *Proc. International Joint Conference on Artificial Intelligence, Montreal* (1995), Morgan Kaufmann, pp. 1985–1991.

[4] LIN, F. Embracing Causality in Specifying the Indeterminate Effects of Actions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-96)* (1996).

[5] LIN, F., AND REITER, R. State Constraints Revisited. *Journal of Logic and Computation 4*, 5 (1994), 655–678. Special Issue on Actions and Processes.

[6] MCCARTHY, J., AND HAYES, P. J. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence* **4**, B. Meltzer and D. Michie, Eds. Edinburgh University Press, Edinburgh, Scotland, 1969, pp. 463–502.

[7] MILLER, R., AND SHANAHAN, M. Narratives in the Situation Calculus. *The Journal of Logic and Computation 4*, 5 (1994), 513–530.

[8] PINTO, J. *Temporal Reasoning in the Situation Calculus.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, Feb. 1994. URL = ftp://ftp.cs.toronto.edu/~cogrob/jpThesis.ps.Z.

[9] PINTO, J. Occurrences in the Hypothetical Worlds of the Situation Calculus: Extended Abstract. In *Reasoning about Actions and Planning in Complex Environments, FAPR'96 Workshop. Technical Report AIDA-96-11, Technische Hochschule Darmstadt, Fachbereich Informatik* (1996), U. C. Sigmund and M. Thielscher, Eds., pp. VII–1 – VII–8.

[10] PINTO, J. Occurrences and Narratives as Constraints in the Branching Structure of the Situation Calculus. Submitted to the Journal of Logic and Computation URL = ftp://lyrcc.ing.puc.cl/pub/jpinto/jlc.ps.gz, 1997.

[11] PINTO, J., AND REITER, R. Temporal Reasoning in Logic Programming: A Case for the Situation Calculus. In *Proceedings of the Tenth International Conference on Logic Programming* (Budapest, June 1993), D. S. Warren, Ed., The MIT Press, pp. 203–221.

[12] REITER, R. *The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a completeness result for goal regression.* Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. Academic Press, San Diego, CA, 1991, pp. 359–380.

[13] REITER, R. Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference(KR'96)* (Cambridge, Massachussetts, U.S.A., Nov. 1996).

[14] SHANAHAN, M. A Circumscriptive Calculus of Events. *Artificial Intelligence 77*, 2 (September 1995), 249–284.

[15] SHOHAM, Y. *Reasoning about Change*. The MIT Press, Cambridge, Mass., 1988.

[16] SHOHAM, Y. Nonmonotonic Reasoning and Causation. *Cognitive Science 14* (1990), 213–252.

[17] THIELSCHER, M. Ramification and causality. *Artificial Intelligence 89* (1997), 317–364.