In *Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning*, 115--122, 2005.

TECHNICAL REPORT
R-301
May 2005

Causality and Counterfactuals in the Situation Calculus

Mark Hopkins and Judea Pearl
Department of Computer Science
University of California, Los Angeles
{mhopkins,judea}@cs.ucla.edu

Abstract

Structural causal models offer a popular framework for exploring causal concepts. However, due to their limited expressiveness, structural models have difficulties coping with such concepts as actual (event-to-event) causation. In this paper, we propose a new type of causal model, based on embedding structural considerations in the language of situation calculus. By using situation calculus as a basic language, we leverage its power to express complex, dynamically changing situations and, by relying on structural considerations, we can formulate an effective theory of counterfactuals within the situation-calculus.

1 Introduction

Researchers have recently devoted a good deal of effort to causality. One of the main branches of this research has focused on structural causal models [Pearl, 2000], which have been applied to a diverse range of problems, including applications in statistics, econometrics, and epidemiology. Their principal strength lies in assessing causal relationships between variables (for example, the causal relationship between the level of pesticide use on crop yield).

The problem with using structural causal models is that the language of structural models is simply not expressive enough to capture certain intricate relationships that are important in causal reasoning. The ontological commitment of these models is to facts alone, assignments of values to random variables, much like propositional logic. Just as propositional logic is not a particularly effective tool for reasoning about dynamic situations, it is similarly difficult to express dynamic situations (with objects, relationships, and time) in terms of structural causal models.

One problem that suffers from this weak expressiveness is termed "actual cause" [Halpern and Pearl, 2001], the identification of the actual cause of an event in a given scenario. For example:

Firing Squad: There are two rifleman (R_1 and R_2) in a firing squad. On their captain's order, they both shoot simultaneously and accurately. The prisoner dies.

From this story, we can ask causal queries such as: did R_1's shot cause the prisoner's death? We can also ask whether the captain's order caused the prisoner's death. The difficulties in defining such a concept of causation have been well-documented in the philosophical literature. In [Halpern and Pearl, 2001], Halpern and Pearl propose a sophisticated definition for actual causality based on structural causal models, however although this definition works on many previously problematic examples, it still does not fit with intuition on all examples. Some of these difficulties can be traced to the limited expressiveness of the structural model formulation. [Hopkins and Pearl, 2003] provides a detailed account of the drawbacks of defining actual cause in the structural model framework.

Hence, we will propose a causal model based on situation calculus, a second-order logical language that has been widely employed to describe dynamic situations [McCarthy, 1963]. Related efforts have been made along these lines [Costello and McCarthy, 1999; Lin, 1995; Gustafsson and Doherty, 1996; Thielscher, 1999]. Our work differs from these by exploiting the valuable lessons that can be learned from the structural model approach. In this paper, we will make steps towards unifying the disparate areas of situation calculus and structural equation modeling by proposing a model-theoretic approach to counterfactuals in the situation calculus. We will also discuss extensions to probabilistic counterfactuals.

2 Review of Structural Causal Models

One of the main benefits of structural causal models is their ability to handle counterfactual queries, i.e. queries asking whether A would have occurred had B not occurred.

Formally, a *causal model* is a triple (U, V, F), in which U is a finite set of random variables (called background or exogenous), V is a finite set of endogenous random variables (disjoint from U), and $F = \{F_X | X \in V\}$ where F_X is a function $Dom(R) \rightarrow Dom(X)$ that assigns a value to X for each setting of the remaining variables in the model $R = U \cup V \backslash \{X\}$. For each X, we can define PA_X, the *parent set* of X, to be the set of variables in R that can affect the value of X (i.e. are non-trivial in F_X). We also assume that the domains of the random variables are finite.

Causal models can be depicted as a *causal diagram*, a directed graph whose nodes correspond to the variables in $U \cup V$ with an edge from Y to $X \in V$ iff $Y \in PA_X$.

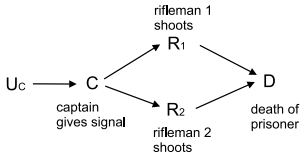

Figure 1: Causal model for the Firing Squad scenario. All variables are propositional. $C = U_C$; $R_1 = C$; $R_2 = C$; $D = R_1 \vee R_2$.

Example In Fig. 1, we see the firing squad scenario expressed as a causal model. Here, $U = \{U_C\}$ and $V = \{C, R_1, R_2, D\}$. All variables are propositional, with value 1 indicating a true proposition, and value 0 indicating that the proposition is false.

If we assume a particular value for the background variables U, then the resulting causal model is called a *causal world*. We generally assume that any particular value for U uniquely determines the values of the variables in V. This always happens when the causal diagram is acyclic (such causal models are called *recursive*). Causal worlds are of interest since they represent a specific situation, while causal models represent a more general scenario. For instance, if we assume that $U_C = 1$ in our firing squad causal model, then the resulting causal world describes our story (given in the introduction). The more general model allows for the situation in which the captain does not signal.

A *probabilistic causal model* gives us a distribution over causal worlds. A probabilistic causal model is a pair $< M, P(u) >$, where M is a causal model, and $P(u)$ is a probability function defined over the domain of U.

To handle counterfactual queries, we define the concept of *submodels*. Given a causal model $M = (U, V, F)$, $X \subseteq V$, $x \in Dom(X)$, the *submodel* of M under *intervention* $X = x$ is $M_{X=x} = (U, V, F_{X=x})$, where $F_{X=x} = \{F_Y | Y \in V \backslash X\} \cup \{X = x\}$. Intuitively, the submodel fixes the values of the variables in X at x. Consequently, the values of the remaining variables represent what values they *would have had* if X had been x in the original model. $M_{X=x}$ and $F_{X=x}$ are typically abbreviated M_x and F_x. The value of variable $Y \in V$ in submodel M_x (under context u) is represented as $Y_{M_x}(u)$ (or simply $Y_x(u)$).

Example (interventions) Consider the firing squad causal model under context $u = \{U_C = 1\}$ and the question: would the prisoner be dead if we *make sure* that R_1 does not fire his gun? This corresponds to evaluating $D_{R_1=0}(u)$. In this case, the captain still signals, so rifleman 2 still shoots. Thus $D_{R_1=0}(u) = 1$, and we conclude that the prisoner still dies in this counterfactual scenario.

Unfortunately, when we attempt to use structural models for the broader question "Did R_1's shot cause the prisoner's death?", we run into difficulties. [Halpern and Pearl, 2001]'s definition answers many such queries correctly, but fails on some. Let us consider various drawbacks with the structural causal model approach when considering event-to-event causation.

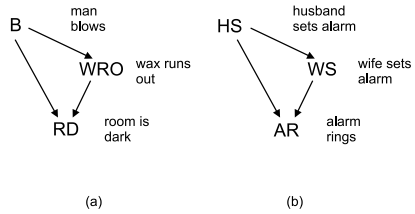

Figure 2: (a) Structural causal model for the Candle scenario. $B = 1$; $WRO = \neg B$; $RD = B \vee WRO$. (b) Structural causal model for the Alarm scenario. $HS = 1$; $WS = \neg HS$; $AR = HS \vee WS$. All variables are endogenous and propositional. Notice that although these models are isomorphic, we would desire them to yield different answers for certain causal queries.

Inability to distinguish between condition and transition: In a structural causal model, everything is represented as a random variable. Thus, one cannot distinguish between an enduring condition (e.g. the man is dead) versus a transitional event (e.g. the man dies). To see why making this distinction can be important when making causal determinations, consider an example where a man dies of a heart attack. We can immediately say that the heart attack is the cause of the man's death (transitional event). However, what if we know for certain that 5 minutes later, the man would have been shot and killed? Is it then accurate to say that the heart attack is the cause of the state of the man being dead 10 minutes after his heart attack? To take this example to the extreme, is it correct to say that the heart attack is the cause of the man *being dead* in the year 3000? Imagine somebody in the year 3000 saying the following: "That man's heart attack is the reason why he is dead today." This suggests the usefulness of making a distinction between conditions and transitions. Consider the following story:

Candle: A man blows out a candle. The candle only had 5 minutes of wax left to burn at the time.

Question: is the man blowing out the candle the cause of darkness in the room one hour later? The intuitive answer is no, since the candle would have burned out by then regardless. However, when modeled as a structural model in Fig. 2(a), Halpern and Pearl's definition concludes that the man's action is indeed the cause of the room being dark one hour after the fact. For contrast, suppose we had the following situation:

Alarm: A man sets his alarm clock before bed. When his wife goes to bed, she goes to set it (for the same time), but realizes it has already been set, so she does nothing. The alarm goes off at the arranged time.

Notice that the structural model for this story (depicted in Fig. 2(b)) is isomorphic to the model for the candle story. Yet the intuitive answer in this case is that the man setting his alarm did indeed cause his alarm to go off. Why the difference? In the first story, the effect in question is an enduring condition, while in the second story, the effect is a transitional event. Such semantic differences are critical when making causal determinations. Notice that in the candle example, we would indeed want to say that the man's action caused the

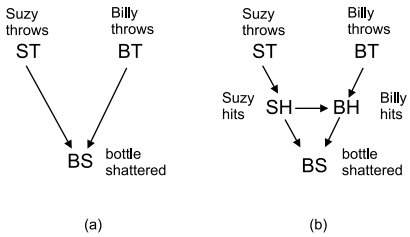

Figure 3: Structural causal models for the Bottle scenario. (a) $ST = 1$; $BT = 1$; $BS = ST \vee BT$. (b) $ST = 1$; $BT = 1$; $SH = ST$; $AR = BT \wedge \neg SH$; $AR = SH \vee BH$. All variables are endogenous and propositional.

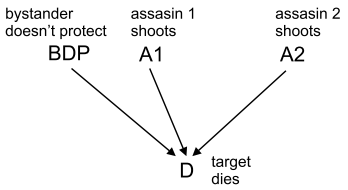

Figure 4: Causal model for the Bystander scenario. All variables are propositional. $BDP = 1$; $A1 = 1$; $A2 = 1$; $D = (BDP \wedge A1) \vee A2$. Notice that BDP and $A1$ are logically indistinguishable.

candle *to extinguish*, which is a transitional event. However, when dealing with enduring conditions, like the room being dark, we generally apply a stricter counterfactual criterion to decide questions of causation. The structural framework cannot make such distinctions, and clearly any definition of causation would have to return identical answers for any query regarding the preceeding two isomorphic causal models.

Inability to express the notion of an object: A primary benefit of a first-order logical language is the ability to reason using objects. Consider the following story from [Hall, 2004]:

Bottle: Suzy and Billy both pick up rocks and throw them at a bottle. Suzy's rock gets there first, shattering the bottle. Both throws were perfectly accurate.

[Halpern and Pearl, 2001] models this story using the causal models depicted in Fig. 3. Under its definition of actual cause, Suzy and Billy's throws both cause the bottle to be shattered in the first model, while only Suzy's throw causes the bottle to be shattered in the corrected second model. Although the model in this case can be corrected to yield the intuitive result, this example demonstrates the potential utility of explicitly representing the objects of the scenario. For example, we can provide a more elegant depiction of the scenario by simply representing the general rule that a rock hitting the bottle causes it to break, and by having distinct objects representing the two rocks. For a more complex scenario, this may be the only practical way for a human to correctly model the situation. Additionally, we may be able to derive information from the general rule that will help us in making causal determinations.

Inability to distinguish between presence and absence of an event: A structural causal model does not make a distinction between an event's occurrence and non-occurrence (all are represented as indistinguishable value assignments to a variable). Nevertheless, when making causal determinations, human beings often take such factors into consideration. Consider the following:

Bystander: Say that two assasins simultaneously shoot a man, and that either shot would have been sufficient to kill him. We would conclude that both assasin's shots are causes of the man's death. Now, suppose that a bystander could have thrown himself into the path of one of the bullets, but did not.

Intuitively, would we conclude that the bystander's inaction was the cause of the man's death? We would not, since the man would have been shot and killed regardless by the other bullet. Nevertheless, given random variables A1 (for the first assasin's shot) and BDP (for BystanderDoesn'tProtect), we can easily see that these variables would be treated identically in the resulting structural causal model, depicted in Fig. 4. Still, we would like different answers when asking whether the first's assasin's shot or the bystander's inaction was a cause of the target's death. The lesson here is that factors such as action versus inaction are important when making causal determinations, and the structural model framework does not make such distinctions.

Hence, let us attempt to go beyond the ontological commitment of structural causal models. In the proceeding sections, we will show how to embed structural causal models in the language of situation calculus.

3 Review of Situation Calculus

The situation calculus is a second-order representation language for dynamically changing worlds. For a formal description of $L_{sitcalc}$, see [Reiter, 2000] – for the purposes of this paper, it suffices to review the following aspects:

Objects: As in first-order logic, we have variables that represent objects in the world. For example, R_1 could represent the first rifleman.

Actions: All changes to the world are brought about by actions. Actions are represented by function symbols and may be parameterized. For example, $shoot(X, Y)$ could represent the action of object X shooting at object Y.

Situations: A situation is a possible world history. Specifically, a situation is simply a sequence of actions. The initial situation is denoted by the distinguished constant S_0. There is also a reserved binary function symbol *do*. $do(\alpha, s)$ represents the situation that results from performing action α in situation s. An example situation could be the following: $do(die(P), do(shoot(R_1, P), do(signal(C), S_0)))$, which represents the situation that results from C giving a signal, followed by A shooting at P, followed by P's death. Notice that above, the actions occur right-to-left as we read it. As a shorthand, we can describe the same situation as: $[signal(C), shoot(R_1, P), die(P)]$. In this form, the starting situation S_0 is assumed.

Relational Fluents: Relational fluents are situation-dependent predicates. They are represented by predicate symbols whose last argument is a situation term. For example, $dead(x, s)$ could be a fluent describing whether object x is

dead in situation s.

Functional fluents: Functional fluents are situation-dependent functions. They are represented by function symbols whose last argument is a situation term. For example, $distance(x, y, s)$ could be a fluent returning the distance between x and y in situation s.

Situation-independent predicates: We will also need predicates with are situation-independent. For example, $rifleman(x)$ could be a predicate returning whether object x is a rifleman.

We should also draw attention to a special binary predicate symbol $\sqsubseteq$: $situation \times situation$ that defines an ordering relation on situations. The semantic interpretation of $s \sqsubseteq s'$ is that s is a subhistory of s'. In other words, s is some subsequence of the actions of s' starting from S_0. For example, $[shoot(R_1, P)] \sqsubseteq [shoot(R_1, P), die(P)]$, but $[die(P)] \not\sqsubseteq [shoot(R_1, P), die(P)]$.

Once we have established a set of objects, actions, fluents, and predicates, we then need a way to describe the dynamics of the world. Specifically, we need to describe the initial state of the world, the conditions under which actions are permissible, and the effect of actions on the world. This is accomplished with three distinct sets of axioms.

Initial Database Axioms: An *initial database axiom* is a first-order situation calculus sentence that is uniform in S_0 (i.e. mentions no situation term except S_0). These axioms are used to describe the initial state of the system, and may include sentences that contain no reference at all to a situation, e.g. $rifleman(R_1)$.

Action Precondition Axioms: An *action precondition axiom* describes the conditions that must hold in order for an action to be permissible in a given situation. For example, $Poss(shoot(x, y), s) \equiv rifleman(x) \wedge sawSignal(x, s)$ states that it is possible for object x to shoot object y in situation s iff x is a rifleman and has seen a signal to fire.

Successor State Axioms: Successor state axioms are used to describe how fluents change in response to actions. For example, $dead(x, do(a, s)) \equiv dead(x, s) \vee a = die(x)$ asserts that object x will be dead after action a if and only if it is already dead in situation s or if a is x's death.

We will refer to our collection of objects, actions, fluents, predicates, and axioms as the *situation-calculus specification* of our problem.

Clearly, not all situations of a given situation-calculus specification are possible, since an action's precondition axiom may not be satisfied when we attempt to execute it. We refer to the "possible" situations as *executable*, and define the concept formally as: $executable(s) = (\forall a, s^*)\ do(a, s^*) \sqsubseteq s \supset Poss(a, s^*)$. In words, a situation is executable if (and only if) it is possible to perform all of its actions in order (from S_0).

4 A Motivating Example

In this section, we will begin to address the issue of translating structural causal models into the language of situation calculus. We will approach this problem by asking the question: what should structural causal models look like in situation calculus? Before proceeding to general formalisms, let us first translate the firing squad causal model, in order to provide motivation.

Objects:
- R1, R2(riflemen)
- C(captain), P(prisoner)

Actions:
- signal(x). x signals.
- shoot(x,y). x shoots y.
- die(x). x dies.

Situation-independent predicates:
- rifleman(x), captain(x), prisoner(x)

Fluents:
- wantsToSignal(x,s).
  (x wants to signal in situation s).
- sawSignal(x,s).
  (x has seen the signal in situation s).
- shot(x,s).
  (x is shot (injured) in situation s).
- dead(x,s).
  (x is dead in situation s).

Action-precondition axioms:
- Poss(signal(x),s) captain(x) wantsToSignal(x,s)
- Poss(shoot(x,y),s) rifleman(x) prisoner(y) sawSignal(x,s)
- Poss(die(x),s) shot(x,s) dead(x,s)

Successor-state axioms:
- wantsToSignal(x,do(a,s)) wantsToSignal(x,s)
- sawSignal(x,do(a,s)) y[a=signal(y)] sawSignal(x,s)
- shot(x,do(a,s)) y[a=shoot(y,x)] shot(x,s)
- dead(x,do(a,s)) a=die(x) dead(x,s)

Initial database axioms:
- rifleman(R1), captain(R1), prisoner(R1), etc.
- (x) (sawSignal(x,S0) shot(x,S0) dead(x,S0))
- wantsToSignal(C), wantsToSignal(R1), etc.

Figure 5: Sitcalc specification for Firing Squad scenario.

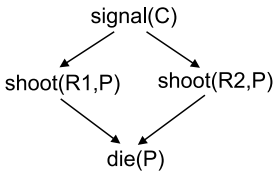

Figure 6: DAG representing time dependencies between actions in the Firing Squad scenario.

We can begin with a situation calculus specification for the scenario. Figure 5 depicts a possible sitcalc specification for the firing squad scenario. How does such a specification compare with the corresponding structural causal model? The key difference is that while the sitcalc specification tells us when actions are allowable, it does not enforce a deterministic progression of events. Specifically, while the sitcalc specification of the firing squad scenario tells us the conditions under which a rifleman *may* fire, it does not state (as the structural model does) that Rifleman A *will* fire if the captain signals. The sitcalc specification thus takes a more general view than we want. In order to model the progression of events in the actual scenario, we need to add more to the sitcalc specification. We need to specify the situation (out of the many permissible situations) that we are currently interested in.

In the firing squad scenario, we are interested in the situation that results from four actions. We can graphically depict these actions as a directed acyclic graph (DAG), such as the one in Fig. 6. The arrows in the graph represent time dependency between the actions, e.g. $signal(C)$ occurs before $shoot(R1, P)$. Such a diagram corresponds to the set of all situations that follow the partial order that it specifies. We can choose any of these situations (it does not make a difference which one we choose) and call it the *potential situation*. Now we can show that the sitcalc specification, together with this situation, enforces a deterministic progression of events.

To extract the actual course of events in the scenario, we try

each action of the potential situation in order. In our example, we try to execute $signal(C)$. We can (its action precondition axiom is satisfied). Then we can try to execute $shoot(R1, P)$. We can also do this. In fact, we can execute each action of our potential situation in order, hence we conclude that the actual course of events in the scenario are that the captain signals, the two riflemen shoot, and the prisoner dies.

The next issue is handling counterfactuals. Say we want to extract the course of events when the captain does not signal, yet the second rifleman shoots anyway. For this, we remove $signal(C)$ from the potential situation, and when we attempt to execute $shoot(R2, P)$, we do so automatically, regardless of whether its action precondition axiom is satisfied. In this case, when we try to execute each action of the potential situation in order, we cannot execute $shoot(R1, P)$ (its action precondition axiom is not satisfied), then we automatically execute $shoot(R2, P)$, and we are able to execute $die(P)$. Thus we conclude that in the counterfactual scenario, the second rifleman shoots and the prisoner dies.

Hence, we see (at least intuitively) how notions of deterministic progression of events and counterfactuals can be modeled with a sitcalc specification and a potential situation. The last issue is how probabilistic causal models are mapped into this framework. The key question here is: what is the correspondent of exogenous variables in the situation calculus framework? The solution is the initial database axioms. Just as exogenous variables specify the given background conditions for structural causal models, so do the initial database axioms in situation calculus. Rather than taking the initial values of the fluents as given, we can put a probability distribution over the possible values of the fluents. For the firing squad scenario, the only uncertain background variable determines whether the captain signals. Hence, all of the probability mass would be distributed among two sets of initial database axioms, differing only by the value assigned to the $wantsToSignal(C)$ fluent.

5 Formalisms

Given the motivating example from the previous section, let us proceed to formalize the notion of a situation calculus causal model. We begin with some preliminary notation.

Definition 1 *Let $S = [a_1, ..., a_n]$ be a situation, and let A be a set of ground action terms. We define S-A as S_{S-A}, constructed as follows:*

1. $S_{S-A} \leftarrow S_0$.
2. *For $i = 1$ to n: if $a_i \notin A$ then $S_{S-A} \leftarrow do(a_i, S_{S-A})$*

Definition 2 *Let S be a situation, and let a be an action. We define $a \in S$ to mean that $(\exists s)do(a, s) \sqsubseteq S$. Similarly, $a \notin S$ means that $(\nexists s)do(a, s) \sqsubseteq S$.*

Intuitively, $S - A$ is the situation S with any mention of the actions in A removed. $a \in S$ means that the action a appears in the sequence of actions S (recall that a situation is simply a sequence of actions).

Recall from the previous section that our causal model will consist of a sitcalc specification (possibly with an incompletely specified initial situation), and a potential situation. Hence, we need to formally define *potential situation*.

Definition 3 *Given a situation-calculus specification D, a* potential situation *is a pair (S, F), where S is a (not necessarily executable) situation of D, and $F : \{a|a \in S\} \rightarrow \{0, 1\}$ is a function that maps each $a \in S$ to either 0 or 1.*

F associates either a zero or a one to each action of the situation. These values are flags to ignore the preconditions of a particular action (as in the example when we force the second rifleman to shoot, regardless of the captain's signal). Generally when initially defining a potential situation, we want F to be the trivial function that maps all actions of S to 0 (signifying that none of them are being forced to occur). We will call this function $F_0(S)$. We are now ready to define causal models in the situation calculus.

Definition 4 *A* situation calculus causal model *(SCCM) is a pair (D, P), where D is a situation calculus specification (possibly with an incompletely specified initial situation) and P is a potential situation of D.*

Given an SCCM $M = (D, P)$, we can fully specify the initial situation with a set U of axioms that fully specify the values of the fluents of D. We shall use $M(U)$ to denote the new SCCM $(D \cup U, P)$.

An SCCM with a fully-specified initial situation would be the SCCM correspondent of a causal world. We can also define a probabilistic SCCM as a pair $< M, P(U) >$, where M is an SCCM, and $P(U)$ is a probability distribution over sets of axioms that fully specify the initial situation of M.

To handle counterfactuals in SCCM $M = (D, P = (S, F))$, we can define a new causal model $[\neg a_1, ..., \neg a_m, b_1, ..., b_n]M$ where $a_i, b_i \in S$. The new model will be the pair $(D, P' = (S', F'))$, where $S' = S - \{a_1, ..., a_m\}$ and $F' : \{a|a \in S'\} \rightarrow \{0, 1\}$ maps the actions of S' to $\{0, 1\}$ such that $F'(a) = 1$ iff $F(a) = 1$ or $(\exists j)(a = b_j)$. Intuitively, we remove the actions from P that we want to force not to occur, and we flag those actions of P that we want to make sure to happen, regardless of their preconditions.

Now we introduce the key operator that extracts the actual course of events in a particular causal model.

Definition 5 *Given a situation calculus causal model $M = (D, P = ([a_1, ..., a_n], F))$ (with a fully specified initial situation), the* natural executable subsituation $NES(M)$ *is constructed iteratively as follows:*

1. $NES(M) \leftarrow S_0$.
2. *For $i = 1$ to n: if ($Poss(a_i, NES(M))$ or $F(a_i) = 1$) then $NES(M) \leftarrow do(a_i, NES(M))$.*

Essentially, we attempt to perform each action of the potential situation in order. If it is possible to perform the action at each given step (or if the action is being forced to occur), we add it to the natural executable subsituation; if not, we omit it. Semantically, the natural executable subsituation of a causal model corresponds to the actual course of events under that model. Notice that M must have a fully specified initial situation, so that the action preconditions can be successfully evaluated.

Given the preceding definition of an SCCM, we can construct a logical language for talking about counterfactuals.

Syntax: We begin with a sitcalc specification D, and define our language with respect to D. We wish to formulate three types of queries: whether a relational fluent holds, whether a functional fluent has a certain value, and whether an action occurs. Thus, we define a *query term* as any ground action term of D, or any ground fluent term of D, with the situation argument removed. For example, for our firing squad sitcalc specification, $dead(P)$ would be a query term, rather than $dead(P, s)$. Define $Q(D)$ as the set of all possible query terms of D.

Now define a *basic causal formula* of our language as one of the form $[\neg a_1, ..., \neg a_m, b_1, ..., b_n]q(U)$, where a_i, b_i are actions of D, $q \in Q(D)$, and U is a set of axioms uniform in S_0 that, together with D, completely specify the initial situation (i.e. all ground fluents are assigned truth values for S_0). Our language $L(D)$ is the set of Boolean combinations of basic causal formulas.

Semantics: A formula in $L(D)$ is true or false in SCCM $M = (D, P)$. We write $M \models \varphi$ if causal formula φ is true in M. For a query term $q = R(X_1, ..., X_k)$ where R is a relational fluent of D, we have $M \models [\neg a_1, ..., \neg a_m,\ b_1, ..., b_n]q(U)$ iff $D \cup U \models R(X_1, ..., X_k, NES([\neg a_1, ..., \neg a_m,\ b_1, ..., b_n]M(U)))$. Similarly for a query term $q = G(X_1, ..., X_k)$ where G is a functional fluent of D, we have $M \models [\neg a_1, ..., \neg a_m,\ b_1, ..., b_n]q(U)$ iff $D \cup U \models G(X_1, ..., X_k,\ NES([\neg a_1, ..., \neg a_m,\ b_1, ..., b_n]M(U)))$. Finally, for a query term $q = a$ where a is a ground action term of D, we have $M \models [\neg a_1, ..., \neg a_m,\ b_1, ..., b_n]q(U)$ if $a \in NES([\neg a_1, ..., \neg a_m,\ b_1, ..., b_n]M(U))$.

The truth value of Boolean combinations of causal formulas are defined in the natural way. We say $M \models \varphi_1 \wedge \varphi_2$ if $M \models \varphi_1$ and $M \models \varphi_2$. Also, $M \models \neg\varphi$ if $\neg(M \models \varphi)$.

Once we have this language for expressing counterfactuals in an SCCM, an extension for probabilistic counterfactuals easily follows. Namely, $Pr([\neg a_1, ..., \neg a_m,\ b_1, ..., b_n]q) = \Sigma_U(Pr(U) * [\neg a_1, ..., \neg a_m,\ b_1, ..., b_n]q(U))$. The probability of a causal formula is the sum of the probabilities of the U (complete specifications of the initial situation) in which the formula holds.

Example For our firing squad example, we have an SCCM $M = (D, P)$, where D is the sitcalc specification in Fig. 5 (without the axiom $wantsToSignal(C, S_0)$), and $P = [signal(C), shoot(R1, P),\ shoot(R2, P), die(P)]$. Let us evaluate the formula $[\neg signal(C), shoot(R2, P)]dead(P)(U)$, where U is the singleton set with axiom $wantsToSignal(C, S_0)$ (notice again the correspondence with exogenous variables in the structural causal model framework). $M \models [\neg signal(C),\ shoot(R2, P)]dead(P)(U)$ if $dead(P, NES([\neg signal(C),\ shoot(R2, P)]M(U)))$ holds. Indeed it does, since $NES([\neg signal(C), shoot(R2, P)]M(U)) = [shoot(R2, P), die(P)]$. Thus we conclude that if the captain does not signal, and yet the second rifleman still (somehow) shoots, the prisoner will still be dead when all is said and done. We can also consider probabilistic counterfactuals. Take probabilistic SCCM $< M, P(U) >$, where M is the SCCM defined above, and $P(U)$ is a distribution over the two singleton sets $\{wantsToSignal(C, S_0)\}$ and $\{\neg wantsToSignal(C, S_0)\}$. Evaluating $Pr([\neg shoot(R2, P)]\ dead(P)) = \Sigma_U([\neg shoot(R2, P)]\ dead(P)(U) * Pr(U))$, we see that the probability that the prisoner ends up dead when the second rifleman is *forced* not to shoot is equivalent to the probability of $wantsToSignal(C, S_0)$, i.e. the prior probability that the captain wants to signal.

	Structural causal model element	SCCM equivalent
1.	Exogenous variables	Initial database axioms
2.	Endogenous variables (enduring conditions)	Fluents
3.	Endogenous variables (transitional events)	Actions
4.	Functional mechanisms (for enduring conditions)	Successor-state axioms
5.	Functional mechanisms (for transitional events)	Action precondition axioms + potential situation

Figure 7: Parallels between structural causal models and SCCMs.

Although we have been drawing parallels between the SCCM framework and the structural model framework as we go along, at this point it might be helpful to summarize these correspondences. In Fig. 7, we show the key components of structural causal models, and highlight their analogs in terms of situation calculus causal models.

6 Further Examples

In this section, we revisit two simple examples which were discussed in Section 2, and show how their representation as SCCMs can provide the expressivity needed to resolve the problems they face in the structural framework. Notice that for questions of actual cause, we are usually concerned exclusively with a specific causal world, hence in the following examples, all sitcalc specifications will have completely specified initial situations.

Example Let us formulate the candle story as an SCCM. The sitcalc specification D (with a completely specified initial situation) is given in Fig. 8. Our potential situation will be $P = (S = [blow(A, C), extinguish(C), waxRunsOut(C)], F_0(S))$ (simply the possible events of our scenario in temporal order). This gives rise to SCCM $M = (D, P)$. Recall that the problem with using the structural model approach was that we could not distinguish between the act of extinguishing (transitional event) and the state of being extinguished (enduring condition). In the SCCM representation however, this is not a problem, since transitional events and enduring conditions are represented as two different classes of object: actions and fluents, respectively. Although providing a definition of actual cause in the SCCM framework is outside of the scope of this paper, it might be instructive to show how certain queries about this model look in the logical representation developed in the previous section. For example, to ask whether the candle would have been manually extinguished had A not blown when he did, we ask

Objects:
- A(man)
- C(candle)

Actions:
- blow(x,y). (x blows at y).
- extinguish(x). (x extinguishes).
- waxRunsOut(x). (x runs out of wax).

Fluents:
- burning(x,s).
(x is burning in situation s).
- blownAt(x,s).
(x is being blown at in situation s).

Situation-independent predicates:
- human(x), candle(x).

Action-precondition axioms:
- Poss(blow(x,y),s) ≡human(x) ∧ candle(y)
- Poss(extinguish(x),s) ≡candle(x) ∧ blownAt(x,s)
- Poss(waxRunsOut(x),s) ≡candle(x)

Successor-state axioms:
- burning(x,do(a,s)) ≡ ¬[a=extinguish(x) ∨ a=waxRunsOut(x)] ∧ burning(x,s)
- blownAt(x,do(a,s)) ≡ ∃y[a=blow(y,x)] ∨ blownAt(x,s)

Initial database axioms:
- human(A), ¬candle(A), ¬human(A), candle(C)
- burning(C,S0), ¬burning(A,S0), (∀x)¬blownAt(x,S0)

Figure 8: Sitcalc specification for the Candle scenario.

Objects:
- A1, A2(assasins), B1, B2(bullets)
- S(bystander), T(target)

Actions:
- shoot(x,y,z). (x shoots y at z).
- stop(x,y). (x stops y).
- die(x). (x dies).

Fluents:
- inFlight(x,y,s).
(x is en route to y in situation s).

Situation-independent predicates:
- human(x), bullet(x), heroic(x).

Action-precondition axioms:
- Poss(shoot(x,y,z),s) ≡human(x) ∧ bullet(y) ∧ human(z)
- Poss(stop(x,y),s) ≡ human(x) ∧ bullet(y) ∧ heroic(x) ∧ (∃z)inFlight(y,z,s)
- Poss(die(x),s) ≡ (∃y)[inFlight(y,x,s) ∧ bullet(y) ∧ human(x)]

Successor-state axioms:
- inFlight(x,y,do(a,s)) ≡ (∃z)[a=shoot(z,x,y)] ∨ ¬(∃z)[a=stop(z,x) ∧ inFlight(x,y,s)]

Initial database axioms:
- human(A1), ¬bullet(A1), etc.
- (∀x)(¬heroic(x)), (∀x)(∀y)(¬inFlight(x,y,S0))

Figure 9: Sitcalc specification for the Bystander scenario.

$M \models [\neg blow(A,C)]extinguish(C)()$. To ask whether the candle would have been burning at the end of the situation (i.e. if the room would have been dark), had A not blown when he did, we ask $M \models [\neg blow(A,C)]burning(C)()$. The reader can verify can the former query evaluates to false, while the latter query evaluates to true, using the tools of the previous section. The key idea here is that since transitional events and enduring conditions are two distinct types of object, a definition of actual cause in the SCCM framework can exploit this distinction, while a definition of actual cause in the structural model approach cannot.

Example Now let us formulate the bystander story as an SCCM. We have $M = (D, P)$, where D is the sitcalc specification given in Fig. 9, and $P = (S = [shoot(A1, B1, T), shoot(A2, B2, T), stop(S, B1), die(T)], F_0(S))$. The difficulty with the structural model approach in this example is that it does not discriminate between the occurrence and non-occurrence of an event (both are indistinguishable value assignments to a random variable), a semantic distinction that can prove extremely helpful in making causal determinations (see Section 2). In the SCCM framework, a fundamental difference is drawn between an action occurring and not occurring. Namely, the action appears in the NES when it occurs, and does not otherwise, e.g. $stop(S, B1) \notin NES(M)$, indicating its non-occurrence. A definition of actual cause in the SCCM framework can be designed to take advantage of this distinction. Another lesson to be drawn here is the utility of objects. Using this formulation of the story, it is meaningful to say things like "The first assasin's bullet reached the man and killed him." A definition of actual cause can potentially exploit the extra information gleaned from knowing *whose* bullets penetrated the man's body.

7 Conclusion

Structural causal models are excellent tools for many types of causality-related questions. Nevertheless, their limited expressivity render them less than ideal for some of the more delicate causal queries, like actual causation. These queries require a language that is suited for dealing with complex, dynamically changing situations. Here, we have outlined a new type of causal model, based on the language of situation calculus. We have developed the notions of a causal model in this language, and drawn parallels between our model and its structural analog. With this new approach, we can leverage the expressive power of situation calculus in attempts to capture complex concepts of causation that have not yet been successfully defined. The next step is to successfully define actual cause within this framework and to demonstrate compatibility with intuition. We believe the framework shows promise in this regard.

Acknowledgements

This work was supported by MURI grant N00014-00-1-0617.

References

[Costello and McCarthy, 1999] Tom Costello and John McCarthy. Useful counterfactuals. *Linkoping Electronic Articles in Computer and Information Science*, 3(2), 1999.

[Gustafsson and Doherty, 1996] J. Gustafsson and P. Doherty. Embracing occlusion in specifying the indirect effects of actions. *KR*, 1996.

[Hall, 2004] Ned Hall. Two concepts of causation. In *Causation and Counterfactuals*, pages 225–276, Cambridge, MA, 2004. MIT Press.

[Halpern and Pearl, 2001] Joseph Halpern and Judea Pearl. Causes and explanations: A structural-model approach – part i: Causes. *UAI*, pages 411–420, 2001.

[Hopkins and Pearl, 2003] Mark Hopkins and Judea Pearl. Clarifying the usage of structural models for commonsense causal reasoning. In *Proceedings of the AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, pages 83–89, Menlo Park, CA, 2003. AAAI Press.

[Lin, 1995] F. Lin. Embracing causality in specifying the indirect effect of actions. *IJCAI*, pages 1985–1991, 1995.

[McCarthy, 1963] John McCarthy. Situations, actions, and causal laws. Technical report, Stanford University, 1963.

[Pearl, 2000] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[Reiter, 2000] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2000.

[Thielscher, 1999] M. Thielscher. From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem. *Artifi cial Intelligence*, 111:277–299, 1999.