A Logic of Actual Cause for Nondeterministic Domains

Maryam Rostamigiv[1], Shakil M. Khan[1], Yves Lespérance[2], and Mriana Yadkoo[3]

[1] University of Regina, Regina, Saskatchewan, Canada
{maryam.rostamigiv,shakil.khan}@uregina.ca
[2] York University, Toronto, Ontario, Canada
lesperan@eecs.yorku.ca
[3] University of Windsor, Windsor, Ontario, Canada
yadkoom@uwindsor.ca

Abstract. Crucial to the process of reasoning is the formalization of the actual causes of observed effects. While there has been a lot of work on root cause analysis, most studies have focused mainly on deterministic domains. In this paper, we build on previous work on actual causation in the situation calculus to deal with causation in nondeterministic domains. We use the nondeterministic situation calculus as our base framework for this. We introduce the notions of "CertainlyCauses" and "PossiblyCauses" that enable the representation of actual cause when the agent does not have any control on and does not know the choices that are made by the environment. We then investigate how regression in the situation calculus can be used to reason about these notions.

Keywords: Actual Cause · Nondeterministic Situation Calculus · Logic.

1 Introduction

The term *causation* refers to a collection of closely-related important philosophical problems dealing with causes and their effects that has been studied since the time of Aristotle. Determining the "actual" causes of an observed effect, which are events chosen from a recorded history of actions that occurred prior to the observation of the effect (also known as the scenario) is one such problem (called "efficient" cause in Aristotelian lingo) that has been extensively researched. Motivated by David Hume's philosophical work and Herbert Simon's early contributions, Pearl [20, 21], and later Halpern [8–10], Halpern and Pearl [11], and others [5, 12, 13, 17] developed computational formalizations of this problem within Structural Equations Models (SEM). While their inspirational work significantly advanced this field, their approach based on SEM has been nevertheless criticized for its limited expressiveness [12, 13, 7].

In response to these criticisms, in recent years researchers have become increasingly interested in studying causation within more expressive action-theoretic frameworks, in particular in that of the situation calculus [1, 2, 16].

Among other things, this allows one to formalize causation from the perspective of individual agents by defining a notion of epistemic causation [14] and by supporting causal reasoning about conative effects, which in turn has proven useful for explaining agent behaviour using causal analysis [15] and has the potential for defining important concepts such as responsibility and blame [23].

While there has been a lot of work on actual causation, the vast majority of the work in this area has focused on deterministic domains. However, a distinguishing feature of the real world is that change is often unpredictable. Very few studies address causation in non-deterministic systems, and those that do, are formalized in SEM-based causal models that are known to have limited expressiveness and suffer from a variety of problems. For instance, recently in [3], Beckers presented an extension of causal models to deal with non-determinism. However, despite improving on expressivity of causal models, it is not clear how one can formalize various aspects of action-theoretic/dynamic frameworks there, e.g. non-persistent change supported by fluents, possible dependency between events, temporal order of event occurrence, etc.

In this paper we build on previous work on actual causation in the situation calculus [2, 14] to deal with causation in nondeterministic domains. We use the nondeterministic situation calculus [4] as our base framework. We introduce notions of "CertainlyCauses" and "PossiblyCauses" that enable the representation of actual cause when the agent does not have any control on and does not know the choices that are made by the environment. We then investigate how regression in the situation calculus can be used to reason about these notions.

The paper is structured as follows. In the next section, we provide an overview of the situation calculus and non-deterministic situation calculus (NDSC) and introduce our running example. In Section 3, we examine the definition of actual cause proposed earlier. In Section 4, we present our definitions for causes in the NDSC. In Section 5, we demonstrate how causes can be simplified into regressable formulae. Finally, we conclude with some discussion in Section 6.

2 Preliminaries

Situation Calculus (SC). The situation calculus is a well-known second-order language for representing and reasoning about dynamic worlds [19, 22]. In the SC, all changes are due to named actions, which are terms in the language. Situations represent a possible world history resulting from performing some actions. The constant S_0 is used to denote the initial situation where no action has been performed yet. The distinguished binary function symbol $do(a, s)$ denotes the successor situation to s resulting from performing the action a. The expression $do([a_1, \cdots, a_n], s)$ represents the situation resulting from executing actions $a_1, \cdots, a_n$, starting with situation s. As usual, a relational/functional fluent representing a property whose value may change from situation to situation takes a situation term as its last argument. There is a special predicate $Poss(a, s)$ used to state that action a is executable in situation s. Also, the special binary predicate $s \sqsubset s'$ represents that s' can be reached from situation s by executing some sequence of actions. $s \sqsubseteq s'$ is an abbreviation of $s \sqsubset s' \vee s = s'$. $s < s'$ is an abbreviation of $s \sqsubset s' \wedge \mathit{Executable}(s')$, where $\mathit{Executable}(s)$ is defined as $\forall a', s'.\, do(a', s') \sqsubseteq s \supset \mathit{Poss}(a', s')$, i.e. every action performed in reaching situation s was possible in the situation in which it occurred. $s \leq s'$ is an abbreviation of $s < s' \vee s = s'$.

In the SC, a dynamic domain is specified using a basic action theory (BAT) $\mathcal{D}$ that includes the following sets of axioms: (i) (first-order or FO) initial state axioms $\mathcal{D}_{S_0}$, which indicate what was true initially; (ii) (FO) action precondition axioms $\mathcal{D}_{ap}$, characterizing $\mathit{Poss}(a, s)$; (iii) (FO) successor-state axioms $\mathcal{D}_{ss}$, indicating precisely when the fluents change; (iv) (FO) unique-names axioms $\mathcal{D}_{una}$ for actions, stating that different action terms represent distinct actions; and (v) (second-order or SO) domain-independent foundational axioms Σ, describing the structure of situations [18]. Although the SC is SO, Reiter [22] showed that for certain type of queries ϕ, $\mathcal{D} \models \phi$ iff $\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models \mathcal{R}[\phi]$, where $\mathcal{R}$ is a syntactic transformation operator called *regression* and $\mathcal{R}[\phi]$ is a SC formula that compiles dynamic aspects of the theory $\mathcal{D}$ into the query ϕ. Thus reasoning in the SC for a large class of interesting queries can be restricted to entailment checking w.r.t. a FO theory [22].

Nondeterministic Situation Calculus (NDSC). An important limitation of the standard SC and BATs is that atomic actions are deterministic. De Giacomo and Lespérance (DL21) [4] proposed a simple extension of the framework to handle nondeterministic actions while preserving the solution to the frame problem. For any primitive action by the agent in a nondeterministic domain, there can be a number of different outcomes. (DL21) takes the outcome as being determined by the agent's action and the environment's reaction to this action. This is modeled by having every action type/function $A(\overrightarrow{x}, e)$ take an additional environment reaction parameter e, ranging over a new sort *Reaction* of environment reactions. The agent cannot control the environment reaction, so it performs the reaction-suppressed version of the action $A(\overrightarrow{x})$ and the environment then selects a reaction e to produce the complete action $A(\overrightarrow{x}, e)$. We call the reaction-suppressed version of the action $A(\overrightarrow{x})$ an *agent action* and the full version of the action $A(\overrightarrow{x}, e)$ a *system action.*

We represent nondeterministic domains using action theories called Nondeterministic Basic Action Theories (NDBATs), which can be seen as a special kind of BAT, where (1) every agent action takes an environment reaction parameter; (2) for each agent action we have an agent action precondition formula: $\mathit{Poss}_{ag}(a(\overrightarrow{x}), s) \overset{\text{def}}{=} \phi_a^{agPoss}(\overrightarrow{x}, s)$; (3) for each agent action we have a reaction independence requirement, stating that the precondition for the agent action is independent of any environment reaction: $\forall e.\, \mathit{Poss}(a(\overrightarrow{x}, e), s) \supset \mathit{Poss}_{ag}(a(\overrightarrow{x}), s)$; (4) for each agent action we also have a reaction existence requirement, stating that if the precondition of the agent action holds then there exists a reaction to it which makes the complete system action executable, i.e., the environment cannot prevent the agent from performing an action when its agent action precondition holds: $\mathit{Poss}_{ag}(a(\overrightarrow{x}), s) \supset \exists e.\, \mathit{Poss}_{ag}(a(\overrightarrow{x}, e), s)$. The above requirements *must* be entailed by the action theory for it to be an NDBAT.

An NDBAT $\mathcal{D}$ is the union of the following disjoint sets: including (1) foundational axioms, (2) unique-names axioms for actions, (3) axioms describing the initial situation, (4) successor-state axioms indicating how fluents change after system actions, and (5) system action precondition axioms, indicating when system actions can occur; while these axioms generally follow the form: $Poss(a(\overrightarrow{x}, e), s) \equiv \phi_a^{Poss}(\overrightarrow{x}, e, s)$, in practice, these axioms often take the form: $Poss(a(\overrightarrow{x}, e), s) \equiv Poss_{ag}(a(\overrightarrow{x}), s) \wedge \psi^{Poss}(\overrightarrow{x}, e, s)$, where $Poss_{ag}(a(\overrightarrow{x}), s)$ denotes conditions necessary for the agent action $a(\overrightarrow{x})$ to occur and $\phi_a^{Poss}(\overrightarrow{x}, e, s)$ captures additional conditions influenced by the environment's response.

Projection and Executability. In the NDSC, executing an agent action in a situation may result in different situations and outcomes depending on the environment reaction. To capture this, (DL21) [4] introduced the defined predicate $Do_{ag}(\overrightarrow{a}, s, s')$, meaning that the system may reach situation s' when the agent executes the sequence of agent actions $\overrightarrow{a}$ starting in situation s depending on environment reactions:

$$Do_{ag}(\epsilon, s, s') \overset{\text{def}}{=} s = s' \text{ (where } \epsilon \text{ is the empty sequence of actions)},$$
$$Do_{ag}([A(\overrightarrow{x}), \sigma], s, s') \overset{\text{def}}{=} \exists e.\ Poss(A(\overrightarrow{x}, e), s) \wedge Do_{ag}(\sigma, do(A(\overrightarrow{x}, e), s), s').$$

A condition ϕ may hold after some executions of a sequence of agent actions $\overrightarrow{a}$ starting in situation s, denoted by $PossiblyAfter(\overrightarrow{a}, \phi, s)$, or it may hold after all executions of $\overrightarrow{a}$ in s, denoted by $CertainlyAfter(\overrightarrow{a}, \phi, s)$. Formally:

$$CertainlyAfter(\overrightarrow{a}, \phi, s) \overset{\text{def}}{=} \forall s'.\ Do_{ag}(\overrightarrow{a}, s, s') \supset \phi[s'],$$
$$PossiblyAfter(\overrightarrow{a}, \phi, s) \overset{\text{def}}{=} \exists s'.\ Do_{ag}(\overrightarrow{a}, s, s') \wedge \phi[s'].$$

Two different notions of executability of $\overrightarrow{a}$ are also defined (see [4]).

Example. Our running example involves a robot navigating between different locations and communicating. We take communication to be subject to interference and assume that the robot can communicate at a given location if the location is not risky and it has not become vulnerable. The robot can move between locations if they are connected and communicate from current location (represented using agent actions $move(i, j)$ and $comm(i)$). While moving to a location, the robot faces the possibility of becoming vulnerable if that location is a risky one. Thus the agent action $move(i, j)$ is associated with the system action $move(i, j, e)$, where the environment reaction e can be either Vul (for becoming vulnerable) or $NotVul$ (for not becoming vulnerable). The communicate agent action on the other hand has only one successful environment reaction and so it is associated with the system action $comm(i, e)$, where $e = Success$.

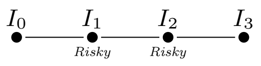

Fig. 1. Interconnections between locations

The precondition axioms for these agent and system actions are as follows.

(1) $Poss_{ag}(move(i,j),s) \overset{\text{def}}{=} At(i,s) \land Connected(i,j)$,

(2) $Poss_{ag}(comm(i),s) \overset{\text{def}}{=} \neg Vul(s) \land \neg Risky(i,s)$,

(1′) $Poss(move(i,j,e),s) \equiv Poss_{ag}(move(i,j),s)$
$\land (Risky(j,s) \supset (e = Vul \lor e = NotVul)) \land (\neg Risky(j,s) \supset e = NotVul)$,

(2′) $Poss(comm(i,e),s) \equiv Poss_{ag}(comm(i),s) \land (e = Success)$.

The fluents in this example are $Vul(s)$, which denotes that the robot is vulnerable in situation s, $At(i,s)$, which states that the robot is at location i in s, and $Risky(i,s)$, which indicates that the location i is risky in s. Certain locations are risky initially and they remain the only risky ones when actions are performed. Also, we have a non-fluent $Connected(i,j)$ to indicate that there is a path from location i to location j.

The successor-state axioms (SSA) for these fluents are as follows:

(3) $At(j, do(a,s)) \equiv \exists i,e.\ a = move(i,j,e)$
$\lor (At(j,s) \land \forall j', e'.\ \neg(a = move(j,j',e')))$,

(4) $Vul(do(a,s)) \equiv \exists i,j.\ a = move(i,j,Vul) \lor Vul(s)$,

(5) $Risky(i, do(a,s)) \equiv Risky(i,s)$.

We also have the following initial state axioms:

$$(6)\ \neg Vul(S_0),\ (7)\ At(I_0,S_0),\ (8)\ Risky(i,S_0) \equiv i = I_1 \lor i = I_2.$$

Finally, there are 4 locations I_0 to I_3 in this domain, and the interconnections between these locations are given by the following axiom (see Fig. 1):

(9) $Connected(i,j) \equiv (i = I_0 \land j = I_1) \lor (i = I_1 \land j = I_2) \lor (i = I_2 \land j = I_3) \lor$
$(i = I_1 \land j = I_0) \lor (i = I_2 \land j = I_1) \lor (i = I_3 \land j = I_2)$.

We will call this NDBAT $\mathcal{D}_1$.

3 Actual Achievement Cause in the Situation Calculus

Given a history of actions/events (often called a scenario) and an observed effect, *actual causation* involves figuring out which of these actions are responsible for bringing about this effect.[4] When the effect is assumed to be false before the

[4] We use actions and events interchangeably.

execution of the actions in the scenario and true afterwards, the notion is referred to as *achievement (actual) causation*. Based on Batusov and Soutchanski's original proposal [2], Khan and Lespérance (KL) recently offered a definition of achievement cause in the SC [14]. Both of these frameworks assume that the scenario is a linear sequence of actions, i.e. no concurrent actions are allowed. KL's proposal can deal with epistemic causes and effects; e.g., an agent may analyze the cause of some newly acquired knowledge, and the cause may include some knowledge-producing action, e.g. *inform*. They showed that an agent may or may not know all the causes of an effect, and can even know some causes while not being sure about others.

To formalize reasoning about effects, KL [14] introduced the notion of *dynamic formulae*. An effect φ in their framework is thus a dynamic formula.[5] Given an effect φ, the actual causes are defined relative to a *narrative* (variously known as a *scenario* or a *trace*) s. When s is a ground situation, the tuple $\langle \mathcal{D}, s, \varphi \rangle$ is often called a *causal setting* [2]. Also, it is assumed that s is executable, and φ was false before the execution of the actions in s, but became true afterwards, i.e. $\mathcal{D} \models \mathit{Executable}(s) \wedge \neg\varphi[S_0] \wedge \varphi[s]$. Here φ is a formula with its situation arguments suppressed and $\varphi[s]$ denotes the formula obtained from φ by restoring the given situation argument s into all fluents in φ (see Def. 2).

Note that since all changes in the SC result from actions, the potential causes of an effect φ are identified with a set of action terms occurring in s. However, since s might include multiple occurrences of the same action, one also needs to identify the situations where these actions were executed. To deal with this, KL required that each situation be associated with a timestamp, which is an integer for their theory. Since in the context of knowledge, there can be different epistemic alternative situations (possible worlds) where an action occurs, using timestamps provides a common reference/rigid designator for the action occurrence. KL assumed that the initial situation starts at timestamp 0 and each action increments the timestamp by one. Thus, their action theory includes the following axioms:

$$\begin{aligned}&\mathit{timeStamp}(S_0) = 0,\\ &\forall a, s, ts.\ \mathit{timeStamp}(do(a, s)) = ts \equiv \mathit{timeStamp}(s) = ts - 1.\end{aligned}$$

With this, causes in their framework is a non-empty set of action-timestamp pairs derived from the trace s given φ.

The notion of *dynamic formulae* is defined as follows:

Definition 1. *Let $\overrightarrow{x}$, θ_a, and $\overrightarrow{y}$ respectively range over object terms, action terms, and object and action variables. The class of* dynamic formulae *φ is defined inductively using the following grammar:*

$$\varphi ::= P(\overrightarrow{x}) \mid \mathit{Poss}(\theta_a) \mid \mathit{After}(\theta_a, \varphi) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists \overrightarrow{y}.\ \varphi.$$

[5] While KL also study epistemic causation, we restrict our discussion to objective causality only.

That is, a dynamic formula (DF) can be a situation-suppressed fluent, a formula that says that some action θ_a is possible, a formula that some DF holds after some action has occurred, or a formula that can built from other DF using the usual connectives. Note that φ can have quantification over object and action variables, but we cannot have quantification over situations or mention the ordering over situations (i.e. $\sqsubset$). We will use φ for DFs.

$\varphi[\cdot]$ is defined as follows:

Definition 2.

$$\varphi[s] \overset{\text{def}}{=} \begin{cases} P(\overrightarrow{x}, s), & \text{if } \varphi \text{ is } P(\overrightarrow{x}) \\ Poss(\theta_a, s), & \text{if } \varphi \text{ is } Poss(\theta_a) \\ \varphi'[do(\theta_a, s)], & \text{if } \varphi \text{ is } After(\theta_a, \varphi') \\ \neg(\varphi'[s]), & \text{if } \varphi \text{ is } (\neg\varphi') \\ \varphi_1[s] \wedge \varphi_2[s], & \text{if } \varphi \text{ is } (\varphi_1 \wedge \varphi_2) \\ \exists \overrightarrow{y}.\ (\varphi'[s]), & \text{if } \varphi \text{ is } (\exists \overrightarrow{y}.\ \varphi'). \end{cases}$$

We will now present KL's definition of causes in the SC. The idea behind how causes are computed is as follows. Given an effect φ and scenario s, if some action of the action sequence in s triggers the formula φ to change its truth value from false to true relative to $\mathcal{D}$, and if there are no actions in s after it that change the value of φ back to false, then this action is an actual cause of achieving φ in s. Such causes are referred to as *primary* causes:

Definition 3 (Primary Cause [14]).

$$\begin{aligned} CausesDirectly(a, ts, \varphi, s) \overset{\text{def}}{=} &\ \exists s_a.\ timeStamp(s_a) = ts \wedge (S_0 < do(a, s_a) \leq s) \\ &\wedge \neg\varphi[s_a] \wedge \forall s'.(do(a, s_a) \leq s' \leq s \supset \varphi[s']). \end{aligned}$$

That is, a executed at timestamp ts is the *primary cause* of effect φ in situation s iff a was executed in a situation with timestamp ts in scenario s, a caused φ to change its truth value to true, and no subsequent actions on the way to s falsified φ.

Now, note that a (primary) cause a might have been non-executable initially. Also, a might have only brought about the effect conditionally and this context condition might have been false initially. Thus earlier actions in the trace that contributed to the preconditions and the context conditions of a cause must be considered as causes as well. The following definition captures both primary and indirect causes:[6]

[6] In this, we need to quantify over situation-suppressed DF. Thus we must encode such formulae as terms and formalize their relationship to the associated SC formulae. This is tedious but can be done essentially along the lines of [6]. We assume that we have such an encoding and use formulae as terms directly.

Definition 4 (Actual Cause [14]).

$$\begin{aligned} \mathit{Causes}(a, ts, \varphi, s) \overset{\text{def}}{=} \forall P.[&\forall a, ts, s, \varphi.(\mathit{CausesDirectly}(a, ts, \varphi, s) \supset P(a, ts, \varphi, s)) \\ &\land \forall a, ts, s, \varphi.(\exists a', ts', s'.(\mathit{CausesDirectly}(a', ts', \varphi, s) \\ &\qquad\qquad \land \mathit{timeStamp}(s') = ts' \land s' < s \\ &\qquad\qquad \land P(a, ts, [\mathit{Poss}(a') \land \mathit{After}(a', \varphi)], s') \\ &\qquad\quad \supset P(a, ts, \varphi, s)) \\ &] \supset P(a, ts, \varphi, s). \end{aligned}$$

Thus, *Causes* is defined to be the least relation P such that if a executed at time-step ts directly causes φ in scenario s then (a, ts, φ, s) is in P, and if a' executed at ts' is a direct cause of φ in s, the timestamp of s' is ts', $s' < s$, and $(a, ts, [\mathit{Poss}(a') \land \mathit{After}(a', \varphi)], s')$ is in P (i.e. a executed at ts is a direct or indirect cause of $[\mathit{Poss}(a') \land \mathit{After}(a', \varphi)]$ in s'), then (a, ts, φ, s) is in P. Here the effect $[\mathit{Poss}(a') \land \mathit{After}(a', \varphi)]$ is that a' be executable and φ hold after a'.

The KL formalization of actual causation was formulated for deterministic domains specified by BATs in the situation calculus. However, it can be used directly for nondeterministic domains, for instance domains specified by NDBATs, as long as one focuses on scenarios that involve sequences of system actions, where both the agent actions and the environment reactions are known. This is not surprising as NDBATs are special kinds of BATS and sequences of system actions are essentially situations. We illustrate this in the example below.

Example (Cont'd). Consider the system action sequence σ_1, where $\sigma_1 = do([\mathit{comm}(I_0, \mathit{Success}); \mathit{move}(I_0, I_1, \mathit{NotVul}); \mathit{move}(I_1, I_2, \mathit{Vul}); \mathit{move}(I_2, I_3, \mathit{NotVul})], S_0)$. We are interested in computing the causes of the effect $\varphi_1 = \mathit{Vul}(s)$, i.e. the robot becoming vulnerable, in scenario σ_1. It can be shown that:

Proposition 1 (Causes of φ_1 in σ_1).

$$\begin{aligned} \mathcal{D}_1 \models \neg \mathit{Causes}(\mathit{comm}(I_0, \mathit{Success}), 0, \varphi_1, \sigma_1) \land \mathit{Causes}(\mathit{move}(I_0, I_1, \mathit{NotVul}), 1, \varphi_1, \sigma_1) \\ \land \mathit{Causes}(\mathit{move}(I_1, I_2, \mathit{Vul}), 2, \varphi_1, \sigma_1) \land \neg \mathit{Causes}(\mathit{move}(I_2, I_3, \mathit{NotVul}), 3, \varphi_1, \sigma_1). \end{aligned}$$

Thus, for example, the action $\mathit{move}(I_1, I_2, \mathit{Vul})$ executed at time-stamp 2 is a cause since it directly caused the robot to become vulnerable. Moreover, $\mathit{move}(I_0, I_1, \mathit{NotVul})$ executed at time-stamp 1 can be shown to be an indirect cause of the $\varphi_1 = \mathit{Vul}$. This is because by axioms (1) and (1$'$) the primary cause of moving from location I_1 to I_2 i.e. $\mathit{move}(I_1, I_2, \mathit{Vul})$ is only possible when the robot is at I_1, which in this scenario was brought about by $\mathit{move}(I_0, I_1, \mathit{NotVul})$.

4 Agent Actions as Causes in the NDSC

We now turn our attention to causation in nondeterministic domains. As mentioned, when the scenario is a sequence of system actions where both the agent actions and the environment reactions are specified, we can use the KL formalization presented earlier to reason about actual causation, and identify causes for effects that are system actions containing both agent actions and environment reactions. But in many cases, we would like to consider scenarios that are sequences of agent actions only and where we don't know what the environment reactions were. Moreover, we want to analyse which agent actions were causes of given effects independently of the environment reactions. We address this question in this section.

We start by defining a notion of *nondeterministic causal setting* that generalizes causal settings and reflects the agent's ignorance about the environment's choices.

Definition 5 (Nondeterministic Causal Setting). *A nondeterministic causal setting is a tuple $\langle \mathcal{D}, \overrightarrow{\alpha}, \varphi \rangle$, where $\mathcal{D}$ is an NDBAT, $\overrightarrow{\alpha}$ is a sequence of agent actions representing the nondeterministic scenario, and φ is a dynamic formula such that:*

$$\mathcal{D} \models \neg\varphi[S_0] \wedge \mathit{PossiblyAfter}(\overrightarrow{\alpha}, \varphi, S_0).$$

Thus a scenario in a nondeterministic causal setting (ND setting, henceforth) is modeled using a sequence of agent actions $\overrightarrow{\alpha}$, with the assumption that this sequence was executed starting in S_0, and φ holds in at least one execution of $\overrightarrow{\alpha}$. Note that, since in this paper we are dealing with actual causation, it is assumed that the effect φ was indeed observed, and thus the agent only considers the executions of $\overrightarrow{\alpha}$ that brought about φ to be candidates for the system action history that actually occurred.

As before, in our framework causes are action and timestamp pairs. However, these actions are now agent actions. Also, since each of the agent actions in the scenario can have multiple outcomes, depending on these outcomes, we might sometimes call an agent action a cause and sometimes not a cause. In some cases, an agent action is a cause of an effect for all possible environment choices associated with the actions in the scenario. In general, given a scenario which is a sequence of agent actions, we will get a tree of possible executions, where each branch is the execution produced by a given set of environment reactions to the agent actions in the sequence. In this tree, it might be that only on certain branches a system action associated with an agent action executed at some timestamp is a cause. Additionally, it is also possible that all the system actions associated with this agent action is a cause in their respective branch. Thus, we have to define two notions of actual causes for agent actions in nondeterministic domains, namely *possibly causes* and *certainly causes*:[7]

[7] We allow both agent actions and system actions to be viewed as causes. An additional possibility is to view the environment's choices as causes, for instance, when there are no other ways of achieving an effect but via certain environment reactions. However, the consequences of such a definition is not clear. For instance, is it reasonable to assign responsibility/blame to nature? Rather than engaging in such philosophical questions, in this paper we focus on causes that involve the agent.

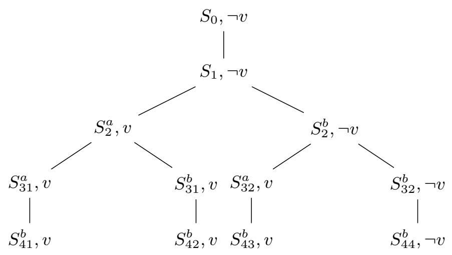

Fig. 2. Executions of agent action sequence $\overrightarrow{\alpha_1}$.

Definition 6 (Possibly Causes). *Let $\langle \mathcal{D}, \overrightarrow{\alpha}, \varphi \rangle$ be an ND setting and $\beta(\overrightarrow{x})$ an agent action in $\overrightarrow{\alpha}$.*

$$\begin{aligned}&\mathit{PossiblyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha}) \overset{\text{def}}{=} \\ &\qquad \exists s.\ \mathit{Do}_{ag}(\overrightarrow{\alpha}, S_0, s) \wedge \varphi[s] \wedge \exists e.\ \mathit{Causes}(\beta(\overrightarrow{x}, e), t, \varphi, s).\end{aligned}$$

That is, agent action $\beta(\overrightarrow{x})$ executed at timestamp t possibly causes φ in scenario $\overrightarrow{\alpha}$ iff there is an execution of $\overrightarrow{\alpha}$ that reaches some situation s, φ holds in s, and for some environment reaction e the associated system action $\beta(\overrightarrow{x}, e)$ executed at timestamp t is a (deterministic) cause of φ in scenario s.

Definition 7 (Certainly Causes). *Let $\langle \mathcal{D}, \overrightarrow{\alpha}, \varphi \rangle$ be an ND setting and $\beta(\overrightarrow{x})$ an agent action in $\overrightarrow{\alpha}$.*

$$\begin{aligned}&\mathit{CertainlyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha}) \overset{\text{def}}{=} \\ &\qquad \forall s.\ \mathit{Do}_{ag}(\overrightarrow{\alpha}, S_0, s) \wedge \varphi[s] \supset \exists e.\ \mathit{Causes}(\beta(\overrightarrow{x}, e), t, \varphi, s).\end{aligned}$$

Thus agent action $\beta(\overrightarrow{x})$ executed at timestamp t certainly causes φ in scenario $\overrightarrow{\alpha}$ iff for all executions of $\overrightarrow{\alpha}$ that reach some situation s in which φ holds, there is an environment reaction e such that the associated system action $\beta(\overrightarrow{x}, e)$ executed at timestamp t is a (deterministic) cause of φ in scenario s. Note that this does not require a system action associated with $\beta(\overrightarrow{x})$ to be a cause in executions s where φ do not hold; this is because since the agent is assumed to have observed the effect φ, such executions can be ruled out as unrealistic, i.e. inconsistent with this assumption.

Example (Cont'd). Consider the agent action sequence $\overrightarrow{\alpha_1}$ below:

$$\overrightarrow{\alpha_1} = \mathit{comm}(I_0); \mathit{move}(I_0, I_1); \mathit{move}(I_1, I_2); \mathit{move}(I_2, I_3).$$

When executed starting on S_0, $\overrightarrow{\alpha_1}$ produces the tree of possible executions shown in Fig. 2. Here, the superscripts a and b represent environment choices *Vul* and *NotVul*, respectively, and $v/\neg v$ indicates whether the agent has become vulnerable or not. Thus for example, in this tree, $S_1 = \mathit{do}(\mathit{comm}(I_0, \mathit{Success}), S_0)$ and $S_2^a = \mathit{do}(\mathit{move}(I_0, I_1, \mathit{Vul}), S_1)$, etc. It is easy to see that the execution of $\overrightarrow{\alpha_1}$ starting in S_0 possibly satisfies $\varphi_1 = \mathit{Vul}()$, e.g. due to the existence of path (S_0, S_{43}^b) in this tree over which the system action sequence in situation σ_1 is executed. Given ND causal setting $\langle \mathcal{D}_1, \overrightarrow{\alpha}_1, \varphi_1 \rangle$, we can in fact show that:

Proposition 2.

$$\begin{aligned}\mathcal{D}_1 \models\; & \neg \mathit{PossiblyCauses}(\mathit{comm}(I_0), 0, \varphi_1, \overrightarrow{\alpha_1}) \\ & \wedge \mathit{CertainlyCauses}(\mathit{move}(I_0, I_1), 1, \varphi_1, \overrightarrow{\alpha_1}) \\ & \wedge \mathit{PossiblyCauses}(\mathit{move}(I_1, I_2), 2, \varphi_1, \overrightarrow{\alpha_1}) \\ & \wedge \neg \mathit{CertainlyCauses}(\mathit{move}(I_1, I_2), 2, \varphi_1, \overrightarrow{\alpha_1}) \\ & \wedge \neg \mathit{PossiblyCauses}(\mathit{move}(I_2, I_3), 3, \varphi_1, \overrightarrow{\alpha_1}).\end{aligned}$$

Thus, $\mathit{comm}(I_0)$ and $\mathit{move}(I_2, I_3)$ do not possibly causes φ_1 because they are not a cause (in the deterministic sense) of φ_1 in any of the executions of $\overrightarrow{\alpha_1}$ depicted in Fig. 2 in which φ_1 holds, i.e. in scenarios S^b_{41}, S^b_{42}, and S^b_{43}. Moreover, $\mathit{move}(I_0, I_1)$ certainly causes (and thus also possibly causes) φ_1; this is because in all the executions of $\overrightarrow{\alpha_1}$ starting in S_0 over which φ_1 holds, it is either a direct cause of φ_1 (as in situations S^b_{41} and S^b_{42}), or it is an indirect cause of φ_1 (as in S^b_{43}). To see the latter, recall Proposition 1 above. Finally, $\mathit{move}(I_1, I_2)$ possibly causes φ_1, but not certainly causes it; this is because there is at least one execution of $\overrightarrow{\alpha_1}$ in which it is a cause of φ_1, e.g. the one ending in S^b_{43}, but not over all executions in which φ_1 holds, e.g. in scenario S^b_{41}.

5 Reasoning about Achievement Causes

We now show that one can use regression to answer queries about various notions of causes under certain conditions. Note that previously KL [14] showed that *Causes* in the SC can be translated into a regressable formula when the situation is ground. DL [4] on the other hand proved that *CertainlyAfter* and *PossiblyAfter* under the right conditions are reducible to regressable formulae. Here we formalize these two techniques and show that it is possible to combine these two to obtain regressable formulae for possibly causes and certainly causes.

We start by defining a new abbreviation $\mathit{Causes}^k(a, t, \varphi, s)$, meaning that action a executed at time t causes φ in situation s in a causal chain of k steps (Definitions 8 and 9 and Theorem 1 were previously informally discussed in [14]).

Definition 8.

$$\mathit{Causes}^k(a, t, \varphi, s) \overset{\text{def}}{=} \begin{cases} \mathit{CausesDirectly}(a, t, \varphi, s), & \text{if } k = 1 \\ \exists a', s', t'.\ \mathit{CausesDirectly}(a', t', \varphi, s) & \\ \quad \wedge\, \mathit{start}(s') = t' \wedge s' < s & \text{if } k > 1. \\ \quad \wedge\, \mathit{Causes}^{k-1}(a, t, [\mathit{Poss}(a') \wedge \mathit{After}(a', \varphi)], s'), & \end{cases}$$

Let us also define $\mathit{Causes}^{\leq k}(a, t, \varphi, s)$, meaning that action a executed at time t causes φ in situation s in a causal chain of at most k steps.

Definition 9.

$$\mathit{Causes}^{\leq k}(a, t, \varphi, s) \overset{\text{def}}{=} \begin{cases} \mathit{Causes}^k(a, t, \varphi, s), & \text{if } k = 1 \\ \mathit{Causes}^k(a, t, \varphi, s) \vee \mathit{Causes}^{\leq k-1}(a, t, \varphi, s), & \text{if } k > 1. \end{cases}$$

Then we can show the following.

Theorem 1.

$$\mathcal{D} \models \mathit{Causes}(a, t, \varphi, do([a_1, \ldots, a_n], S_0) \equiv \mathit{Causes}^{\leq n}(a, t, \varphi, do([a_1, \ldots, a_n], S_0)).$$

Intuitively, this follows from the fact that any causal chain between a_1 and a_n, every cause of φ holding in $do([a_1, \ldots, a_n], s)$ has at most n steps.

We next define a translation function τ^+ that we will use to prove regressability of causes (this is one of our main contributions).

Definition 10.

$$\tau^+(\mathit{CausesDirectly}(b, t, \varphi, do([a_1, \ldots, a_k, \ldots, a_n], S_0))) \stackrel{\text{def}}{=}$$
$$b = a_k \wedge t = k - 1 \wedge \neg\varphi[do([a_1, \ldots, a_{k-1}], S_0)] \wedge \bigwedge_{i=k}^{n} \varphi[do([a_1, \ldots, a_i], S_0)].$$

We extend τ^+ to also include indirect causes.

Definition 11.

$$\tau^+(\mathit{Causes}(a, t, \varphi, do([a_1, \ldots, a_n], S_0))) \stackrel{\text{def}}{=} \tau^+(\mathit{Causes}^{\leq n}(a, t, \varphi, do([a_1, \ldots, a_n], S_0))),$$

where $\tau^+(\mathit{Causes}^{\leq k}(a, t, \varphi, S_0)) \stackrel{\text{def}}{=}$

$$\begin{cases} \tau^+(\mathit{Causes}^{k}(a, t, \varphi, S_0)), & \text{if } k = 1 \\ \tau^+(\mathit{Causes}^{k}(a, t, \varphi, S_0)) \vee \tau^+(\mathit{Causes}^{\leq k-1}(a, t, \varphi, S_0)), & \text{if } k > 1. \end{cases}$$

and $\tau^+(\mathit{Causes}^{k}(a, t, \varphi, do([a_1, \ldots, a_n], S_0))) \stackrel{\text{def}}{=}$

$$\begin{cases} \tau^+(\mathit{CausesDirectly}(a, t, \varphi, do([a_1, \ldots, a_n], S_0))), & \text{if } k = 1 \\ \bigvee_{i=2}^{n} \big[\tau^+(\mathit{CausesDirectly}(a_i, \mathit{start}(do([a_1, \ldots, a_{i-1}], S_0)), \varphi, do([a_1, \ldots, a_n], S_0))) \\ \qquad \wedge\, \tau^+(\mathit{Causes}^{k-1}(a, t, [\mathit{Poss}(a_i) \wedge \mathit{After}(a_i, \varphi)], do([a_1, \ldots, a_i], S_0)))\big], \\ & \text{if } k > 1. \end{cases}$$

Finally, we extend τ^+ to deal with formulae with ground situation terms (here s is a ground situation term and ψ, possibly with decorations, is a formula with ground situation terms):[8]

Definition 12. *Let ψ be a formula where all the situation terms are ground and v be a variable of environment reaction or object sort. Then let:*

$$\tau^+(\psi) \stackrel{\text{def}}{=} \begin{cases} P(\overrightarrow{x}, s), & \text{if } \psi = P(\overrightarrow{x}, s), \\ \mathit{Poss}(a, s), & \text{if } \psi = \mathit{Poss}(a, s), \\ \neg(\tau^+(\psi')), & \text{if } \psi = (\neg\psi'), \\ \tau^+(\psi_1) \wedge \tau^+(\psi_2), & \text{if } \psi = (\psi_1 \wedge \psi_2), \\ \exists v.\ \tau^+(\psi'), & \text{if } \psi = (\exists v.\ \psi'). \end{cases}$$

[8] Note that although we want this definition to deal with, among other things, arbitrary situation-grounded dynamic formulae, we don't need to cover the case where $\psi = \mathit{After}(a, \varphi)[s]$ since this is reduced to $\varphi[do(a, s)]$ by Definition 2.

Using this, we can show the following.

Theorem 2. *Let α be a ground system action term, φ be a dynamic formula without action variables, and σ be a ground situation term. Then we have:*

$$\mathcal{D} \models \mathit{Causes}(\alpha, t, \varphi, \sigma) \equiv \tau^+(\mathit{Causes}(\alpha, t, \varphi, \sigma)).$$

Then, by the theorem, our first major result about regressability of causes immediately follows (this result was previously explored in [14], but without the details of translation τ^+).

Corollary 1. *If σ is a ground situation term, α is a ground action term, and φ does not involve any action variables, then $\mathit{Causes}(\alpha, t, \varphi, \sigma)$ is equivalent to a regressable formula.*

This follows from Theorem 2 and the fact that when the above conditions hold, $\tau^+(\mathit{Causes}(\alpha, t, \varphi, \sigma))$ produces a Boolean combination of action- and situation-grounded *CausesDirectly* formula, which in turn reduces to a regressable formula that talks about whether the associated effect is true or not in different such grounded situations.

We next investigate the regressability of possibly and certainly causes. To this end, we define a couple of translation functions that convert these to regressable formulae. We start by introducing a translation function τ to deal with *CertainlyAfter* and *PossiblyAfter* (the idea behind Definition 13 and Theorem 3 is borrowed from [4]).

Definition 13. *Let β be an agent action function and $\overrightarrow{\alpha}$ be an agent action sequence. Then let:*

$$\tau(\mathit{CertainlyAfter}(\overrightarrow{\alpha}, \psi, s)) \overset{\text{def}}{=}$$
$$\begin{cases} \psi[s], & \text{if } \overrightarrow{\alpha} = \mathit{nil}, \\ \forall e.\ \phi_\beta^{\mathit{Poss}}(\overrightarrow{x}, e, s) \supset \tau(\mathit{CertainlyAfter}(\overrightarrow{\gamma}, \psi, do(\beta(\overrightarrow{x}, e), s))), & \\ & \text{if } \overrightarrow{\alpha} = [\beta(\overrightarrow{x}), \overrightarrow{\gamma}]. \end{cases}$$

$$\tau(\mathit{PossiblyAfter}(\overrightarrow{\alpha}, \psi, s)) \overset{\text{def}}{=}$$
$$\begin{cases} \psi[s], & \text{if } \overrightarrow{\alpha} = \mathit{nil}, \\ \exists e.\ \phi_\beta^{\mathit{Poss}}(\overrightarrow{x}, e, s) \wedge \tau(\mathit{PossiblyAfter}(\overrightarrow{\gamma}, \psi, do(\beta(\overrightarrow{x}, e), s))), & \\ & \text{if } \overrightarrow{\alpha} = [\beta(\overrightarrow{x}), \overrightarrow{\gamma}]. \end{cases}$$

Using this, we can show the following result that previously appeared in [4].

Theorem 3. *Let $\mathcal{D}$ be an NDBAT, $\overrightarrow{\alpha}$ a ground agent action sequence, S a situation term containing no situation or action variables, and ψ a situation suppressed formula containing no action variables. Then $\mathit{CertainlyAfter}(\overrightarrow{\alpha}, \psi, S)$ is equivalent to a regressable formula, and so is $\mathit{PossiblyAfter}(\overrightarrow{\alpha}, \psi, S)$.*

Example (Cont'd). Let us consider the agent action sequence $\overrightarrow{\alpha_2} = \mathit{move}(I_0, I_1); \mathit{move}(I_1, I_2)$, where the robot moves from location I_0 to I_1 and then from I_1 to I_2, starting in S_0. Note that, it is possible that the agent becomes vulnerable after executing $\overrightarrow{\alpha_2}$, i.e. $\mathit{PossiblyAfter}(\overrightarrow{\alpha_2}, \mathit{Vul}, S_0)$. We can show that this can be reduced to the following, which is a regressable formula.

Proposition 3.

$$\begin{aligned}&\mathcal{D}_1 \models \mathit{PossiblyAfter}(\overrightarrow{\alpha_2}, \mathit{Vul}, S_0) \equiv\\ &\quad \exists e.\ \mathit{At}(I_0, S_0) \land \mathit{Connected}(I_0, I_1) \land (e = \mathit{Vul} \lor e = \mathit{NotVul})\\ &\quad \land \exists e'.\ \mathit{At}(I_1, do(move(I_0, I_1, e), S_0)) \land \mathit{Connected}(I_1, I_2) \land (e' = \mathit{Vul} \lor e' = \mathit{NotVul})\\ &\quad \land \mathit{Vul}(do(move(I_1, I_2, e'), do(move(I_0, I_1, e), S_0))).\end{aligned}$$

Next, we show that certainly/possibly causes are equivalent to a formula about certainly/possibly after. Using this, τ, and τ^+, we will then compose a function τ^* to reduce certainly and possibly causes to regressable formulae (what follows is one of our main contributions).

Proposition 4. *Let $\beta(\overrightarrow{x})$ be a ground agent action, $\overrightarrow{\alpha}$ be a ground agent action sequence, and φ be a dynamic formula without action variables. Then we have:*

$$\begin{aligned}&\mathcal{D} \models \mathit{CertainlyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha}) \equiv\\ &\qquad \mathit{CertainlyAfter}(\overrightarrow{\alpha}, \neg\varphi \lor \exists e.\ \mathit{Causes}(\beta(\overrightarrow{x}, e), t, \varphi), S_0).\\ &\mathcal{D} \models \mathit{PossiblyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha}) \equiv\\ &\qquad \mathit{PossiblyAfter}(\overrightarrow{\alpha}, \exists e.\ \mathit{Causes}(\beta(\overrightarrow{x}, e), t, \varphi), S_0).\end{aligned}$$

τ^* is defined as follows.

Definition 14.

$$\begin{aligned}&\tau^*(\mathit{CertainlyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha})) \overset{\text{def}}{=}\\ &\qquad \tau^+(\tau(\mathit{CertainlyAfter}(\overrightarrow{\alpha}, \neg\varphi \lor \exists e.\ \mathit{Causes}(\beta(\overrightarrow{x}, e), t, \varphi)), S_0))),\\ &\tau^*(\mathit{PossiblyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha})) \overset{\text{def}}{=}\\ &\qquad \tau^+(\tau(\mathit{PossiblyAfter}(\overrightarrow{\alpha}, \exists e.\ \mathit{Causes}(\beta(\overrightarrow{x}, e), t, \varphi), S_0))).\end{aligned}$$

Using this, we can show the following results.

Theorem 4. *Let $\beta(\overrightarrow{x})$ be a ground agent action, $\overrightarrow{\alpha}$ be a ground agent action sequence, and φ be a dynamic formula without action variables. Then we have:*

$$\begin{aligned}&\mathcal{D} \models \mathit{CertainlyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha}) \equiv \tau^*(\mathit{CertainlyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha})),\\ &\mathcal{D} \models \mathit{PossiblyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha}) \equiv \tau^*(\mathit{PossiblyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha})).\end{aligned}$$

Finally, we prove our main result about the regressability of causes in the NDSC.

Theorem 5. *Let $\mathcal{D}$ be an NDBAT, $\beta(\overrightarrow{x})$ be a ground agent action, $\overrightarrow{\alpha}$ be a ground agent action sequence, and φ be a dynamic formula without action variables. Then $\mathit{CertainlyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha})$ is equivalent to a regressable formula, and so is $\mathit{PossiblyCauses}(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha})$.*

This follows from Theorem 4 and the definitions of τ^*, τ, and τ^+. Note that $\tau(\cdots)$ on the right-hand sides of Definition 14 will eventually be reduced to formulae that are constructed from regressable sub-formulae as well as *Causes* predicates (see the base cases in Definition 13). τ^+, as specified in Definition 10 to 12, then converts these formulae into regressable formulae.

Example (Cont'd). We can show that $PossiblyCauses(move(I_0, I_1), 0, Vul, \overrightarrow{\alpha_2})$ can be translated into the following regressable formula.

Proposition 5.

$$\begin{aligned}
&\mathcal{D}_1 \models PossiblyCauses(move(I_0, I_1), 0, Vul, \overrightarrow{\alpha_2}) \equiv \\
&\exists e'.\ At(I_0, S_0) \land Connected(I_0, I_1) \land (e' = Vul \lor e' = NotVul) \\
&\ \land \exists e''.\ At(I_1, do(move(I_0, I_1, e'), S_0)) \land Connected(I_1, I_2) \land (e'' = Vul \lor e'' = NotVul) \\
&\ \land ([\neg Vul(S_0) \land Vul(do(move(I_0, I_1, e'), S_0)) \\
&\qquad \land Vul(do(move(I_1, I_2, e''), do(move(I_0, I_1, e'), S_0)))] \lor \\
&\quad\ [\neg Vul(do(move(I_0, I_1, e'), S_0)) \land Vul(do(move(I_1, I_2, e''), do(move(I_0, I_1, e'), S_0))) \\
&\qquad \land \neg\varphi_2(S_0) \land \varphi_2(do(move(I_0, I_1, e'), S_0))]),
\end{aligned}$$

where $\varphi_2 = [Poss(move(I_1, I_2, e'')) \land After(move(I_1, I_2, e''), Vul)]$.

This regressable formula requires that for some environment reaction e', the first move action in the scenario must be executable in S_0, and for another environment reaction e'', the second move action must be executable afterwards. Moreover, $move(I_0, I_1, e')$ executed at time-stamp 0 is required to be either a primary or a secondary cause of the robot becoming vulnerable. The former demands Vul to be false initially in S_0 and to become and remain true after the execution of the first move action till the end of the trace. The latter stipulates that $move(I_1, I_2, e'')$ must be a primary cause of Vul, and thus Vul must have been false before its execution but must become and remain true afterwards. As well, $move(I_0, I_1, e')$ must have caused the preconditions of the latter move action and the conditions under which the latter move action brings about Vul (this indirect effect is represented using φ_2 above), and thus φ_2 must have been false in S_0 but must become and remain true after the first move action was performed until the execution of the next move action.

6 Conclusion

Motivated by the nondeterministic nature of real world action and change, in this paper, we proposed a formalization of actual achievement causes in the nondeterministic situation calculus [4]. We showed that in such domains, when the environment reactions are not known, two different notions of causes can be defined, one where an agent action is a cause for all possible environment reactions, and thus it is certainly a cause, and another where the agent action is a cause for at least one environment choice, i.e. it is possibly a cause. Extending on previous work, we also proved that these notions can be straightforwardly translated to regressable formulae. While our initial results look promising and shed some light on reasoning about these notions, in practice our translations generate large formulae. To deal with this, we are currently looking at how one could produce more compact versions of the translations. We are also looking into the epistemics of causation in nondeterministic domains, extending our previous work in [14].

References

1. Batusov, V., Soutchanski, M.: Situation calculus semantics for actual causality. In: Gordon, A.S., Miller, R., Turán, G. (eds.) Proceedings of the Thirteenth International Symposium on Commonsense Reasoning, COMMONSENSE 2017, London, UK, November 6-8, 2017. CEUR Workshop Proceedings, vol. 2052. CEUR-WS.org (2017)
2. Batusov, V., Soutchanski, M.: Situation calculus semantics for actual causality. In: McIlraith, S.A., Weinberger, K.Q. (eds.) Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018. pp. 1744–1752. AAAI Press (2018)
3. Beckers, S.: Nondeterministic causal models. CoRR **abs/2405.14001** (2024)
4. De Giacomo, G., Lespérance, Y.: The nondeterministic situation calculus. In: Bienvenu, M., Lakemeyer, G., Erdem, E. (eds.) Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021. pp. 216–226 (2021)
5. Eiter, T., Lukasiewicz, T.: Complexity results for structure-based causality. Artificial Intelligence **142**(1), 53–89 (2002). https://doi.org/10.1016/S0004-3702(02)00271-0, https://doi.org/10.1016/S0004-3702(02)00271-0
6. Giacomo, G.D., Lespérance, Y., Levesque, H.J.: Congolog, a concurrent programming language based on the situation calculus. Artificial Intelligence **121**(1-2), 109–169 (2000). https://doi.org/10.1016/S0004-3702(00)00031-X, https://doi.org/10.1016/S0004-3702(00)00031-X
7. Glymour, C., Danks, D., Glymour, B., Eberhardt, F., Ramsey, J.D., Scheines, R., Spirtes, P., Teng, C.M., Zhang, J.: Actual causation: A stone soup essay. Synthese **175**(2), 169–192 (2010). https://doi.org/10.1007/s11229-009-9497-9, https://doi.org/10.1007/s11229-009-9497-9
8. Halpern, J.Y.: Axiomatizing causal reasoning. Journal of Artificial Intelligence Research **12**, 317–337 (2000). https://doi.org/10.1613/jair.648, https://doi.org/10.1613/jair.648
9. Halpern, J.Y.: A modification of the halpern-pearl definition of causality. In: Yang, Q., Wooldridge, M.J. (eds.) Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015. pp. 3022–3033. AAAI Press (2015), http://ijcai.org/Abstract/15/427
10. Halpern, J.Y.: Actual Causality. MIT Press (2016)
11. Halpern, J.Y., Pearl, J.: Causes and explanations: A structural-model approach. part i: Causes. The British Journal for the Philosophy of Science **56**(4), 843–887 (2005)
12. Hopkins, M.: The Actual Cause: From Intuition to Automation. Ph.D. thesis, University of California Los Angeles (2005)
13. Hopkins, M., Pearl, J.: Causality and counterfactuals in the situation calculus. Journal of Logic and Computation **17**(5), 939–953 (2007). https://doi.org/10.1093/logcom/exm048, https://doi.org/10.1093/logcom/exm048
14. Khan, S.M., Lespérance, Y.: Knowing why - on the dynamics of knowledge about actual causes in the situation calculus. In: Dignum, F., Lomuscio, A., Endriss, U., Nowé, A. (eds.) AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021. pp. 701–709. ACM (2021)

15. Khan, S.M., Rostamigiv, M.: On explaining agent behaviour via root cause analysis: A formal account grounded in theory of mind. In: Gal, K., Nowé, A., Nalepa, G.J., Fairstein, R., Radulescu, R. (eds.) ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland. Frontiers in Artificial Intelligence and Applications, vol. 372, pp. 1239–1247. IOS Press (2023)
16. Khan, S.M., Soutchanski, M.: Necessary and sufficient conditions for actual root causes. In: Giacomo, G.D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., Lang, J. (eds.) ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020). Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 800–808. IOS Press (2020). https://doi.org/10.3233/FAIA200169, https://doi.org/10.3233/FAIA200169
17. Leitner-Fischer, F., Leue, S.: Causality checking for complex system models. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7737, pp. 248–267. Springer (2013). https://doi.org/10.1007/978-3-642-35873-9_16, https://doi.org/10.1007/978-3-642-35873-9_16
18. Levesque, H.J., Pirri, F., Reiter, R.: Foundations for the situation calculus. Electronic Transactions on Artificial Intelligence (ETAI) **2**, 159–178 (1998)
19. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. Machine Intelligence **4**, 463–502 (1969)
20. Pearl, J.: On the definition of actual cause. Tech. Rep. R-259, University of California Los Angeles (1998)
21. Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press (2000)
22. Reiter, R.: Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, Cambridge, MA, USA (2001)
23. Yazdanpanah, V., Gerding, E.H., Stein, S., Dastani, M., Jonker, C.M., Norman, T.J., Ramchurn, S.D.: Reasoning about responsibility in autonomous systems: challenges and opportunities. AI Soc. **38**(4), 1453–1464 (2023)