Reasoning About Actual Causes in Nondeterministic Domains

Shakil M. Khan[1], **Yves Lespérance**[2], **Maryam Rostamigiv**[1]

[1]University of Regina, Regina, SK, Canada
[2]York University, Toronto, ON, Canada
shakil.khan@uregina.ca, lesperan@eecs.yorku.ca, maryam.rostamigiv@uregina.ca

Abstract

Reasoning about the causes behind observations is crucial to the formalization of rationality. While extensive research has been conducted on root cause analysis, most studies have predominantly focused on deterministic settings. In this paper, we investigate causation in more realistic nondeterministic domains, where the agent does not have any control on and may not know the choices that are made by the environment. We build on recent preliminary work on actual causation in the nondeterministic situation calculus to formalize more sophisticated forms of reasoning about actual causes in such domains. We investigate the notions of "Certainly Causes" and "Possibly Causes" that enable the representation of actual cause for agent actions in these domains. We then show how regression in the situation calculus can be extended to reason about such notions of actual causes.

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 (Pearl 1998, 2000), and later Halpern (Halpern 2000, 2015, 2016), Halpern and Pearl (Halpern and Pearl 2005), and others (Eiter and Lukasiewicz 2002; Hopkins 2005; Hopkins and Pearl 2007) 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 (Hopkins 2005; Hopkins and Pearl 2007; Glymour et al. 2010), and others have expanded SEM with additional features (Leitner-Fischer and Leue 2013) or proposed alternate formalizations of actual cause, e.g. (Bochman 2018; Beckers and Vennekens 2018; de Lima and Lorini 2024).

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 (Batusov and Soutchanski 2017, 2018; Khan and Soutchanski 2020). Among other things, this allows one to formalize causation from the perspective of individual agents by defining a notion of epistemic causation (Khan and Lespérance 2021) and by supporting causal reasoning about conative effects, which in turn has proven useful for explaining agent behaviour using causal analysis (Khan and Rostamigiv 2023) and has the potential for defining important concepts such as responsibility and blame (Yazdanpanah et al. 2023).

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 nondeterministic 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 Beckers (2024) presented an extension of causal models to deal with nondeterminism. 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.

To deal with this, building on previous work on actual causation in the situation calculus (Batusov and Soutchanski 2018; Khan and Lespérance 2021), more recently (Rostamigiv et al. 2024) formalized actual causes in more expressive nondeterministic action-theoretic domains. They used the nondeterministic situation calculus (De Giacomo and Lespérance 2021) as their base framework. They introduced notions of "Certainly Causes" and "Possibly Causes" that enable the representation of actual cause when the agent does not control her actions' outcomes and does not know the choices that are made by the environment. However, this early work formalizes reasoning about causes in these domains by considering all possible evolutions of the scenario, which makes reasoning computationally intractable.[1] In this

[1]They reason about "Certainly Causes" by considering all the

paper, we build on this preliminary work and formalize more effective forms of reasoning about actual causes in such nondeterministic domains. In particular, we extend regression in the situation calculus to provide a more effective mechanism to reason about actual causes.

The paper is structured as follows. In the next section, we provide an overview of the situation calculus and nondeterministic situation calculus (NDSC) and introduce our running example. Then, we examine the definition of actual cause proposed earlier. After that, we present the definitions of various causal notions in the NDSC. Next, we formalize reasoning about actual causes in the NDSC. In particular, we prove some properties showing how regression in the situation calculus can be extended to deal with these notions. Finally, we conclude the paper with some discussion.

Preliminaries

Situation Calculus (SC). The situation calculus is a well-known second-order language for representing and reasoning about dynamic worlds (McCarthy and Hayes 1969; Reiter 2001). 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 $s \sqsubset s' \wedge Executable(s')$, where $Executable(s)$ is defined as $\forall a', s'.\ do(a', s') \sqsubseteq s \supset Poss(a', s')$, 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 $Poss(a, s)$; (iii) (FO) successor-state axioms $\mathcal{D}_{ss}$, specifying how the fluents change when an action is performed and providing a solution to the frame problem; (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 (Levesque, Pirri, and Reiter 1998).

A key feature of BATs is the existence of a sound and complete *regression mechanism* for answering queries about situations resulting from performing a sequence of actions (Pirri and Reiter 1999; Reiter 2001). In a nutshell, the regression operator $\mathcal{R}^*$ reduces a formula ϕ about a particular future situation to an equivalent formula $\mathcal{R}^*[\phi]$ about the initial situation S_0. A formula ϕ is regressable if and only if (i) all situation terms in it are of the form $do([a_1, \ldots, a_n], S_0)$, (ii) in every atom of the form $Poss(a, \sigma)$, the action function is specified, i.e., a is of the form $A(t_1, \ldots, t_n)$, (iii) it does not quantify over situations, and (iv) it does not contain $\sqsubset$ or equality over situation terms. Thus in essence, a formula is regressable if it does not contain situation variables.

In the following, we define a one-step variant of $\mathcal{R}^*$, $\mathcal{R}$.

Definition 1 (The Regression Operator)
(1) When ϕ is a non-fluent atom, including equality atoms without functional fluents as arguments, or when ϕ is a fluent atom, whose situation argument is S_0, $\mathcal{R}[\phi] = \phi$.
(2a) For a non-functional fluent F, whose successor-state axiom in $\mathcal{D}$ is $F(\overrightarrow{x}, do(a, s)) \equiv \Phi_F(\overrightarrow{x}, a, s)$, $\mathcal{R}[F(\overrightarrow{t}, do(\alpha, \sigma))] = \Phi_F(\overrightarrow{t}, \alpha, \sigma)$.
(2b) For an equality literal with a functional fluent f, whose successor-state axiom is $f(\overrightarrow{x}, do(a, s)) = y \equiv \Phi_f(\overrightarrow{x}, y, a, s)$, $\mathcal{R}[f(\overrightarrow{t}, do(\alpha, \sigma)) = t'] = \Phi_f(\overrightarrow{t}, t', \alpha, \sigma)$.
(2c) For a $Poss$ literal with the action precondition axiom of the form $Poss(A(\overrightarrow{x}), s) \equiv \Pi_A(\overrightarrow{x}, s)$, $\mathcal{R}[Poss(A(\overrightarrow{t}), \sigma)] \equiv \mathcal{R}[\Pi_A(\overrightarrow{t}, \sigma)]$.
(3) For any non-atomic formulae, regression is defined inductively: $\mathcal{R}[\neg\phi] = \neg\mathcal{R}[\phi]$, $\mathcal{R}[\phi_1 \wedge \phi_2] = \mathcal{R}[\phi_1] \wedge \mathcal{R}[\phi_2]$, $\mathcal{R}[\exists v.\ \phi] = \exists v.\ \mathcal{R}[\phi]$.

$\mathcal{R}^*$ can then be defined as the repeated application of $\mathcal{R}$ until further applications leave the formula unchanged.

Another key result about BATs is the relative satisfiability theorem (Pirri and Reiter 1999; Reiter 2001): $\mathcal{D}$ is satisfiable if and only if $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$ is satisfiable (the latter being a purely first-order theory).

Nondeterministic Situation Calculus (NDSC). An important limitation of the standard SC and BATs is that atomic actions are deterministic. De Giacomo and Lespérance (DL21) (De Giacomo and Lespérance 2021) 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*.

possible executions of the scenario, and determining whether the given action was an actual cause in each of them. However, the number of distinct executions of such a scenario/agent action sequence is exponential in the number of nondeterministic actions in the scenario.

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, $Poss_{ag}(a(\overrightarrow{x}), s) \stackrel{\text{def}}{=} \phi_a^{agPoss}(\overrightarrow{x}, s)$; (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 $\forall e.\ Poss(a(\overrightarrow{x}, e), s) \supset Poss_{ag}(a(\overrightarrow{x}), s)$; (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 $Poss_{ag}(a(\overrightarrow{x}), s) \supset \exists e.\ Poss_{ag}(a(\overrightarrow{x}, e), s)$. The above requirements *must* be entailed by the action theory for it to be an NDBAT.

A 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: $Poss(a(\overrightarrow{x}, e), s) \equiv Poss_{ag}(a(\overrightarrow{x}), s) \wedge \varphi^{Poss}(\overrightarrow{x}, e, s)$, 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) 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 $\overrightarrow{a}$ depending on environment reactions:

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

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

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

Two different notions of executability of $\overrightarrow{a}$ are also defined (see (De Giacomo and Lespérance 2021)).

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

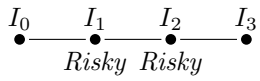

Figure 1: Interconnections between locations

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 system action $comm(i, e)$, where $e = Succ$.

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

(1) $Poss_{ag}(move(i, j), s) \equiv At(i, s) \wedge Connected(i, j)$,
(2) $Poss_{ag}(comm(i), s) \equiv \neg Vul(s) \wedge \neg Risky(i, s)$,
(1′) $Poss(move(i, j, e), s) \equiv Poss_{ag}(move(i, j), s)$
$\wedge (Risky(j, s) \supset (e = Vul \vee e = NotVul))$
$\wedge (\neg Risky(j, s) \supset e = NotVul)$,
(2′) $Poss(comm(i, e), s) \equiv Poss_{ag}(comm(i), s) \wedge e = Succ$.

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 an edge from i to j.

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

(3) $At(j, do(a, s)) \equiv \exists i, e.\ a = move(i, j, e)$
$\vee (At(j, s) \wedge \forall j', e'.\ \neg(a = move(j, j', e'))$,
(4) $Vul(do(a, s)) \equiv \exists i, j.\ a = move(i, j, Vul) \vee Vul(s)$,
(5) $Risky(i, do(a, s)) \equiv Risky(i, s)$.

We also have the following initial state axioms:

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

Finally, there are 4 locations, I_0 to I_3, and the interconnections between these axiomatized using $Connected$ is depicted in Fig. 1. We will call this NDBAT $\mathcal{D}_1$.

Actual Achievement Cause in the SC

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.[2] When the effect is assumed to be false before the

[2]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 (2018) original proposal, Khan and Lespérance (2021) (KL21) recently offered a definition of achievement cause in the SC. Both of these frameworks assume that the scenario is a linear sequence of actions, i.e. no concurrent actions are allowed. (KL21)'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, (KL21) introduced the notion of *dynamic formulae*. An effect φ in their framework is thus a dynamic formula.[3] 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* (Batusov and Soutchanski 2018). 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. $\mathcal{D} \models Executable(s) \wedge \neg\varphi[S_0] \wedge \varphi[s]$. 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. 3 below).

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, (KL21) 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. (KL21) 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:

$$time(S_0) = 0,$$
$$\forall a, s, ts.\ time(do(a, s)) = ts \equiv time(s) = ts - 1.$$

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 2 *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:*

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

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 3

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

We will now present (KL21)'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 4 (Primary Cause (KL21))

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

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:[4]

Definition 5 (Actual Cause (KL21))

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

[3] While (KL21) also study epistemic causation, we restrict our discussion to objective causality only.

[4] 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 (Giacomo, Lespérance, and Levesque 2000). We assume that we have such an encoding and use formulae as terms directly.

Thus, $Causes$ is defined to be the least relation P such that if a executed at timestamp 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 $(a, ts, [Poss(a') \wedge After(a', \varphi)], s')$ is in P (i.e. a executed at ts is a direct or indirect cause of $[Poss(a') \wedge After(a', \varphi)]$ in s'), then (a, ts, φ, s) is in P. Here the effect $[Poss(a') \wedge After(a', \varphi)]$ is that a' be executable and φ hold after a'.

The (KL21) 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 in the scenario σ_1, where

$$\sigma_1 = do([comm(I_0, Succ), move(I_0, I_1, NotVul), \\ move(I_1, I_2, Vul), move(I_2, I_3, NotVul)], S_0).$$

We are interested in computing the causes of the effect $\varphi_1 = Vul$, i.e., the robot becoming vulnerable, in this scenario σ_1. It can be shown that:

Proposition 1 (Causes of φ_1 in σ_1)

$$\mathcal{D}_1 \models \neg Causes(comm(I_0, Succ), 0, \varphi_1, \sigma_1) \\ \wedge Causes(move(I_0, I_1, NotVul), 1, \varphi_1, \sigma_1) \\ \wedge Causes(move(I_1, I_2, Vul), 2, \varphi_1, \sigma_1) \\ \wedge \neg Causes(move(I_2, I_3, NotVul), 3, \varphi_1, \sigma_1).$$

Thus, e.g., the action $move(I_1, I_2, Vul)$ executed at timestamp 2 is a cause since it directly caused the robot to become vulnerable. Moreover, $move(I_0, I_1, NotVul)$ executed at timestamp 1 can be shown to be an indirect cause of the φ_1. This is because by axioms (1) and (1′) the primary cause of moving from location I_1 to I_2 i.e. $move(I_1, I_2, Vul)$ is only possible when the robot is at I_1, which in this scenario was brought about by $move(I_0, I_1, NotVul)$.

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 (KL21) 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 6 (Nondeterministic Causal Setting) *A nondeterministic causal setting is a tuple $\langle \mathcal{D}, \overrightarrow{\alpha}, \varphi \rangle$, where $\mathcal{D}$ is a NDBAT, $\overrightarrow{\alpha}$ is a sequence of agent actions representing the nondeterministic scenario, and φ is a dynamic formula s.t.:*

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

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. Also, φ was initially false and possibly holds after $\overrightarrow{\alpha}$ starting in S_0, i.e. holds after at least one execution of $\overrightarrow{\alpha}$ starting in S_0. Notice that, since we deal with actual causation, which is the problem of determining the causes of an *already observed* effect, just like most previous work on actual causation, we also assume that the effect has already been observed after the (in our case, nondeterministic) actions in the scenario occurred. Thus, any execution after which the effect φ did not occur is excluded from consideration and the agent only considers executions after which φ holds to be candidates for the actual one (as the agent has already observed the effect).

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*:[5]

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

$$PCauses(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha}) \stackrel{\text{def}}{=} \\ \exists s.\ Do_{ag}(\overrightarrow{\alpha}, S_0, s) \wedge \varphi[s] \wedge \exists e.\ Causes(\beta(\overrightarrow{x}, e), t, \varphi, s).$$

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.

[5] 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.

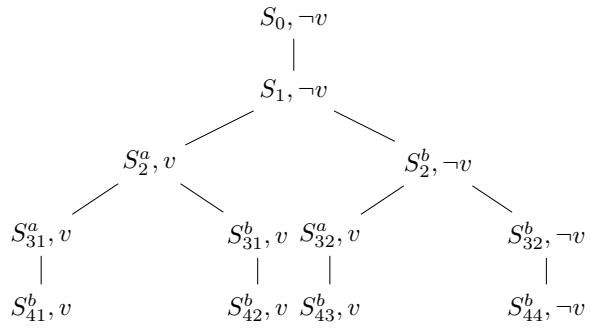

Figure 2: Executions of agent action sequence $\overrightarrow{\alpha_1}$.

Definition 8 (Certainly Causes) *Let $\langle \mathcal{D}, \overrightarrow{\alpha}, \varphi \rangle$ be a ND setting and $\beta(\overrightarrow{x})$ an agent action in $\overrightarrow{\alpha}$.*

$$CCauses(\beta(\overrightarrow{x}), t, \varphi, \overrightarrow{\alpha}) \stackrel{\text{def}}{=}$$
$$\forall s.\ Do_{ag}(\overrightarrow{\alpha}, S_0, s) \wedge \varphi[s] \supset \exists e.\ Causes(\beta(\overrightarrow{x}, e), t, \varphi, s).$$

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 a 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

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

When executed starting in 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, e.g., in this tree $S_1 = do(comm(I_0, Succ), S_0)$ and $S_2^a = do(move(I_0, I_1, Vul), S_1)$, etc. It is easy to see that the execution of $\overrightarrow{\alpha_1}$ starting in S_0 possibly satisfies $\varphi_1 = Vul()$, e.g. due to the existence of path (S_0, S_{43}^b) in this tree over which the system action sequence in 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

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

Thus, $comm(I_0)$ and $move(I_2, I_3)$ 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_{41}^b, S_{42}^b, and S_{43}^b. Moreover, $move(I_0, I_1)$ 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_{41}^b and S_{42}^b), or it is an indirect cause of φ_1 (as in S_{43}^b). To see the latter, recall Proposition 1 above. Finally, $move(I_1, I_2)$ 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_{43}^b, but it is not in all executions in which φ_1 holds, e.g. in S_{41}^b.

Reasoning About Achievement Causes

We now extend regression to answer queries about various notions of causes, and $CAfter$ and $PAfter$. Note that, combining ideas from (KL21) and (DL21), recently in a preliminary work (Rostamigiv et al. 2024) showed that one can obtain equivalent regressable formulae for Possibly Causes and Certainly Causes. However, they do this by translating these into formulae that refer to all possible executions of the scenario/agent action sequence, making reasoning intractable. Here, we develop a more effective approach by extending regression so that Certainly/Possibly Causes in $do(a, s)$ is transformed into an equivalent formula about s. In the following, we prove a series of properties, which will be the basis of our proposed extended regression operator.

Reasoning About System Actions as Causes. We start by showing that $Causes$ in $do(a, s)$ can be reduced to a formula that only mentions $Causes$ in s:

Proposition 3

$$\begin{aligned}\mathcal{D} \models\ & Causes(b, t, \varphi, do(a, s)) \equiv \\ & (time(s) = t \wedge b = a \wedge \neg\varphi[s] \wedge \varphi[do(a, s)]) \vee \\ & (time(s) > t \wedge \varphi[s] \wedge \varphi[do(a, s)] \wedge Causes(b, t, \varphi, s)) \vee \\ & (time(s) > t \wedge \neg\varphi[s] \wedge \varphi[do(a, s)] \\ & \qquad \wedge Causes(b, t, Poss(a) \wedge After(a, \varphi), s)).\end{aligned}$$

Proof Sketch: $\Leftarrow$ For the first disjunct, the result follows immediately from the first implication in the definition of $Causes$. For the third disjunct, it follows immediately from the second implication in the definition of $Causes$. For the second disjunct, the result follows by induction on the number of actions in the causal chain from b to the direct cause of φ holding in $do(a, s)$.

$\Rightarrow$ The result follows by induction on the number of actions in the causal chain from b to the direct cause of φ holding in $do(a, s)$. $\square$

Thus, $Causes(b, t, \varphi, do(a, s))$ can be reduced to one of 3 cases. Either b, which is the same action as a, directly causes φ in $do(a, s)$, and in that case the timestamp of s is t (i.e. a was performed at t), φ was false before the execution of a, and became true afterwards; or b caused φ in an earlier situation and a has no (positive or negative) contributions to this, and in that case the timestamp of s (i.e. the execution time of a) is greater than t, φ was true before and after the execution of a, and b executed at t caused φ in the preceding situation s; or a directly caused φ and b indirectly caused φ by ensuring that the preconditions of a and the context conditions under which a caused φ holds, and in that case the timestamp of s is greater than t, φ was false before the execution of a and became true afterwards, and b executed at t caused the effect $[Poss(a) \wedge After(a, \varphi)]$ in scenario s.

Reasoning About Certainly and Possibly Causes. Next, we show how a $CAfter$ formula (and a $PAfter$ formula) relative to a sequence of agent actions starting in some situation can be reduced to one relative to the shorter sequence obtained by removing the last action from the sequence. This result along with the next will be then used for defining regression for Certainly/Possibly Causes.

Proposition 4

$$\mathcal{D} \models CAfter([\alpha_1, \ldots, \alpha_{n-1}, \alpha_n], \varphi, s) \equiv CAfter([\alpha_1, \ldots, \alpha_{n-1}], CAfter(\alpha_n, \varphi), s) \equiv CAfter([\alpha_1, \ldots, \alpha_{n-1}], \forall e.\ Poss(\alpha_n(e), now) \supset \varphi[do(\alpha_n(e), now)], s).$$

$$\mathcal{D} \models PAfter([\alpha_1, \ldots, \alpha_{n-1}, \alpha_n], \varphi, s) \equiv PAfter([\alpha_1, \ldots, \alpha_{n-1}], PAfter(\alpha_n, \varphi), s) \equiv PAfter([\alpha_1, \ldots, \alpha_{n-1}], \exists e.\ Poss(\alpha_n(e), now) \wedge \varphi[do(\alpha_n(e), now)], s).$$

Notice that the situation argument in the formula inside the context of $CAfter$ (and $PAfter$) above will be provided by the external context once the $CAfter$ (and $PAfter$, resp.) is fully expanded. We often suppress this situation argument; we also sometimes use a placeholder now that stands for it.

Finally, we can show that a $CCauses$ (and $PCauses$) formula can be converted to a certainly after (possibly after, resp.) causes formula.

Proposition 5

$$\mathcal{D} \models CCauses(\beta, t, \varphi, \overrightarrow{\alpha}) \equiv CAfter(\overrightarrow{\alpha}, \neg\varphi \vee \exists e.\ Causes(\beta(e), t, \varphi), S_0).$$

$$\mathcal{D} \models PCauses(\beta, t, \varphi, \overrightarrow{\alpha}) \equiv PAfter(\overrightarrow{\alpha}, \varphi \wedge \exists e.\ Causes(\beta(e), t, \varphi), S_0).$$

The Extended Regression Operator. We extend the set of regressable formulae to include $Causes(b, t, \varphi, do(a, s))$, $CAfter(\overrightarrow{\alpha}, \varphi, s)$, $PAfter(\overrightarrow{\alpha}, \varphi, s)$, $CCauses(\beta, t, \varphi, \overrightarrow{\alpha})$, and $PCauses(\beta, t, \varphi, \overrightarrow{\alpha})$, with the same restrictions on their arguments as imposed in the original definition of regressable formula, as well as $CAfter(\epsilon, \varphi, s)$ and $PAfter(\epsilon, \varphi, s)$, where ϵ is the empty agent action sequence.

We also extend $\mathcal{R}$ to define regression of these additional constructs; regression for the other cases are covered by Definition 1 above. We denote this one-step extended regression operator using $\mathcal{R}_{ext}$. As usual, we use $\mathcal{R}^*_{ext}$ to denote the repeated application of $\mathcal{R}_{ext}$ until further applications leave the argument formula unchanged.

Definition 9 (The Extended Regression Operator $\mathcal{R}_{ext}$)

(1) – (3) When the formula ϕ to be regressed is of the forms discussed in Definition 1, $\mathcal{R}_{ext}(\phi) = \mathcal{R}(\phi)$.
(4) If ϕ is an extended regressable formula of the form $Causes(b, t, \varphi, do(a, s))$, then:

$$\mathcal{R}_{ext}[Causes(b, t, \varphi, do(a, s))] = (time(s) = t \wedge b = a \wedge \neg\varphi[s] \wedge \mathcal{R}_{ext}[\varphi[do(a, s)]]) \vee (time(s) > t \wedge \varphi[s] \wedge \mathcal{R}_{ext}[\varphi[do(a, s)]] \wedge Causes(b, t, \varphi, s)) \vee (time(s) > t \wedge \neg\varphi[s] \wedge \mathcal{R}_{ext}[\varphi[do(a, s)]] \wedge Causes(b, t, Poss(a) \wedge \mathcal{R}_{ext}[\varphi[do(a, s')]]^{-1}, s)).$$

(5) If ϕ is an extended regressable formula that is of the form $CAfter(\overrightarrow{\alpha}, \varphi, s)$ or $PAfter(\overrightarrow{\alpha}, \varphi, s)$ with a possibly empty sequence ϵ of agent actions $\overrightarrow{\alpha}$, then:

$$\mathcal{R}_{ext}[CAfter(\epsilon, \varphi, s)] = \mathcal{R}_{ext}[\varphi[s]].$$
$$\mathcal{R}_{ext}[CAfter([\alpha_1, \ldots, \alpha_{n-1}, \alpha_n], \varphi, s)] = CAfter([\alpha_1, \ldots, \alpha_{n-1}], \forall e.\ Poss(\alpha_n(e)) \supset \mathcal{R}_{ext}[\varphi[do(\alpha_n(e), s')]]^{-1}, s).$$
$$\mathcal{R}_{ext}[PAfter(\epsilon, \varphi, s)] = \mathcal{R}_{ext}[\varphi[s]].$$
$$\mathcal{R}_{ext}[PAfter([\alpha_1, \ldots, \alpha_{n-1}, \alpha_n], \varphi, s)] = PAfter([\alpha_1, \ldots, \alpha_{n-1}], \exists e.\ Poss(\alpha_n(e)) \wedge \mathcal{R}_{ext}[\varphi[do(\alpha_n(e), s')]]^{-1}, s).$$

(6) If ϕ is an extended regressable formula that is of the form $CCauses(\beta, t, \varphi, \overrightarrow{\alpha})$ or $PCauses(\beta, t, \varphi, \overrightarrow{\alpha})$, then:

$$\mathcal{R}_{ext}[CCauses(\beta, t, \varphi, [\alpha_1, \cdots, \alpha_{n-1}, \alpha_n])] = CAfter([\alpha_1, \ldots, \alpha_{n-1}], \forall e'.\ Poss(\alpha_n(e')) \supset \mathcal{R}_{ext}[\phi^*[do(\alpha_n(e'), s')]]^{-1}, S_0),$$
where, $\phi^* = \neg\varphi \vee \exists e.\ Causes(\beta(e), t, \varphi)$.

$$\mathcal{R}_{ext}[PCauses(\beta, t, \varphi, [\alpha_1, \cdots, \alpha_{n-1}, \alpha_n])] = PAfter([\alpha_1, \ldots, \alpha_{n-1}], \exists e'.\ Poss(\alpha_n(e')) \wedge \mathcal{R}_{ext}[\phi^*[do(\alpha_n(e'), s')]]^{-1}, S_0),$$
where, $\phi^* = \varphi \wedge \exists e.\ Causes(\beta(e), t, \varphi)$.

These can be justified directly using Proposition 3, 4, and 5 above. Intuitively, the additional definitions in the extended regression reduce causes in scenario $do(a, s)$ to that in scenario s via reasoning by cases and using the definition of causes and regression;[6] reduce certainly/possibly causes relative to a sequence to that for a shorter sequence, and eventually to a regressable situation calculus formula, which is then regressed;[7] and reduce certainly/possibly causes relative to a (non-empty) sequence of agent actions to a certainly/possibly after formula relative to a shorter sub-sequence, which can then be further reduced using case (5) above.

Having defined this extended regression operator, we are now ready to present our key result:

Theorem 1 *If ϕ is an extended regressable formula and $\mathcal{D}$ is an NDBAT, then:*

$$\mathcal{D} \models \phi \ \textit{iff}\ \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}^*_{ext}[\phi].$$

[6]Notice that the formula inside the context of $Causes$ is situation-suppressed. On the other hand, the regression operator $\mathcal{R}_{ext}$ requires a situation argument. To deal with this, here we introduce a new situation variable s' and use the φ^{-1} operator from (Scherl and Levesque 2003) that suppresses the situation argument of φ by removing the last (situation) argument of all the fluents in φ. Thus, e.g., $\mathcal{R}_{ext}[\varphi[do(\alpha_n(e), s')]]^{-1}$ introduces the situation term $do(\alpha_n(e), s')$ to the situation suppressed formula φ, performs the regression, and then suppresses the situation argument in the result.

[7]Recall that $CCauses$ and $PCauses$ do not take a situation argument, but a sequence of agent actions, which is assumed to be executed starting in S_0. We could have generalized this, however.

The proof of this theorem is similar to that of the regression theorem in the SC (Pirri and Reiter 1999; Reiter 2001), but uses Propositions 3 to 5 for the additional cases.

Example (Cont'd). Consider the agent action sequence $\overrightarrow{\alpha_2} = move(I_0, I_1);\ move(I_1, I_2)$. Note that, $\mathcal{D}_1 \models CCauses(move(I_0, I_1), 0, Vul, \overrightarrow{\alpha_2}, S_0)$. We can show that:

Proposition 6

$$
\begin{aligned}
&\mathcal{R}_{ext}[CCauses(move(I_0, I_1), 0, Vul, \overrightarrow{\alpha_2})] = \\
&CAfter([move(I_0, I_1)], \\
&\quad \forall e'.\ Poss(move(I_1, I_2, e')) \supset \\
&\quad [\neg(e' = Vul \vee Vul) \vee \\
&\quad (\exists e.\ (time > 0 \wedge Vul \\
&\qquad \wedge Causes(move(I_0, I_1, e), 0, Vul)) \vee \\
&\qquad (time > 0 \wedge \neg Vul \wedge e' = Vul \\
&\qquad \wedge Causes(move(I_0, I_1, e), 0, \phi'))))], S_0), \\
&\quad \text{where, } \phi' = Poss(move(I_1, I_2, e')) \wedge (e' = Vul \vee Vul).
\end{aligned}
$$

Repeated applications of $\mathcal{R}_{ext}$ yields a formula about S_0 that can be checked against $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$ in $\mathcal{D}_1$. The complete derivation is available in an extended version; see (Khan, Lespérance, and Rostamigiv 2024).

Conclusion

Motivated by the nondeterministic nature of real world action and change, in this paper, we studied reasoning about actual achievement causes in the NDSC (De Giacomo and Lespérance 2021). 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 showed that one can define a regression operator in the situation calculus to reason about these notions. Note that our definition of extended regression enjoys the benefits of Reiter's regression operator in that it completely sidesteps the second-order part of the theory. Khan and Soutchanski (2020) reported a Prolog implementation of Batusov and Soutchanski's (2018) original proposal, which we think can be extended to deal with the nondeterministic case with some effort.

Here, we focused on actual causation, which assumes that the effect was observed. We thus did not attempt to compute the causes of effects that the agent is unsure about. Such incompleteness must be dealt through the epistemics of causation, where in some possible worlds an agent might consider some effect to be true and some actions to be causes of the effect; in some other possible worlds, she might consider another set of actions to be causes; while in yet other worlds, she might even consider the effect to be false. See (Khan and Lespérance 2021) for an account of this in a deterministic setting. Causal knowledge and its dynamics in the context of nondeterministic actions become even more interesting.

Besides the SEM-based framework discussed in the introduction, another framework that is also closely related to our work is the one proposed by Banihashemi, Khan, and Soutchanski (2022), where the authors define a weak notion of causation, which is intuitively similar to our notion of possibly causes. However, in that paper they primarily focused on abstraction. Also, their semantics is based on the (ordinary) situation calculus, and nondeterminism is handled using nondeterministic ConGolog programs (Giacomo, Lespérance, and Levesque 2000). This makes the semantics quite complicated, in contrast to our simple proposal. They also did not define a notion of certainly causes or address how reasoning about causes can be performed.

As discussed above, perhaps the closest to our work is the preliminary study proposed recently in (Rostamigiv et al. 2024), where the authors also looked at causation in nondeterministic domains. However, as mentioned before, reasoning about causes in that framework is defined using a process of direct translation that in practice generates large formulae. In contrast, we showed how one can extend regression in the situation calculus to produce more compact formulae.

Much work remains. A key restriction in these recent situation calculus-based action-theoretic proposals (including ours) is that these assume that the scenario is simply a sequence of actions, and thus do not allow concurrent actions; we are currently working to address this. We are also looking into the epistemics of causation in nondeterministic domains, extending previous work by Khan and Lespérance (2021).

Acknowledgements

This work is partially supported by the National Science and Engineering Research Council of Canada, by the University of Regina, and by York University.

References

Banihashemi, B.; Khan, S. M.; and Soutchanski, M. 2022. From Actions to Programs as Abstract Actual Causes. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 5470–5478. AAAI Press.

Batusov, V.; and Soutchanski, M. 2017. Situation Calculus Semantics for Actual Causality. In Gordon, A. S.; Miller, R.; and Turán, G., eds., *Proceedings of the Thirteenth International Symposium on Commonsense Reasoning, COMMONSENSE 2017, London, UK, November 6-8, 2017*, volume 2052 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Batusov, V.; and Soutchanski, M. 2018. Situation Calculus Semantics for Actual Causality. In McIlraith, S. A.; and 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*, 1744–1752. AAAI Press.

Beckers, S. 2024. Nondeterministic Causal Models. *CoRR*, abs/2405.14001.

Beckers, S.; and Vennekens, J. 2018. A Principled Approach to Defining Actual Causation. *Synthese*, 195(2): 835–862.

Bochman, A. 2018. Actual Causality in a Logical Setting. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 1730–1736. ijcai.org.

De Giacomo, G.; and Lespérance, Y. 2021. The Nondeterministic Situation Calculus. In Bienvenu, M.; Lakemeyer, G.; and Erdem, E., eds., *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, 216–226.

de Lima, T.; and Lorini, E. 2024. Model Checking Causality. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, 3324–3332. ijcai.org.

Eiter, T.; and Lukasiewicz, T. 2002. Complexity Results for Structure-based Causality. *Artificial Intelligence*, 142(1): 53–89.

Giacomo, G. D.; Lespérance, Y.; and Levesque, H. J. 2000. ConGolog, A Concurrent Programming Language based on the Situation Calculus. *Artificial Intelligence*, 121(1-2): 109–169.

Glymour, C.; Danks, D.; Glymour, B.; Eberhardt, F.; Ramsey, J. D.; Scheines, R.; Spirtes, P.; Teng, C. M.; and Zhang, J. 2010. Actual Causation: A Stone Soup Essay. *Synthese*, 175(2): 169–192.

Halpern, J. Y. 2000. Axiomatizing Causal Reasoning. *Journal of Artificial Intelligence Research*, 12: 317–337.

Halpern, J. Y. 2015. A Modification of the Halpern-Pearl Definition of Causality. In Yang, Q.; and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 3022–3033. AAAI Press.

Halpern, J. Y. 2016. *Actual Causality*. MIT Press. ISBN 978-0-262-03502-6.

Halpern, J. Y.; and Pearl, J. 2005. Causes and Explanations: A Structural-Model Approach. Part I: Causes. *The British Journal for the Philosophy of Science*, 56(4): 843–887.

Hopkins, M. 2005. *The Actual Cause: From Intuition to Automation*. Ph.D. thesis, University of California Los Angeles.

Hopkins, M.; and Pearl, J. 2007. Causality and Counterfactuals in the Situation Calculus. *Journal of Logic and Computation*, 17(5): 939–953.

Khan, S. M.; and Lespérance, Y. 2021. Knowing Why - On the Dynamics of Knowledge about Actual Causes in the Situation Calculus. In Dignum, F.; Lomuscio, A.; Endriss, U.; and Nowé, A., eds., *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, 701–709. ACM.

Khan, S. M.; Lespérance, Y.; and Rostamigiv, M. 2024. Reasoning about Actual Causes in Nondeterministic Domains - Extended Version. *CoRR*.

Khan, S. M.; and Rostamigiv, M. 2023. 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.; and Radulescu, R., eds., *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, 1239–1247. IOS Press.

Khan, S. M.; and Soutchanski, M. 2020. Necessary and Sufficient Conditions for Actual Root Causes. In Giacomo, G. D.; Catalá, A.; Dilkina, B.; Milano, M.; Barro, S.; Bugarín, A.; and 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)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 800–808. IOS Press.

Leitner-Fischer, F.; and Leue, S. 2013. Causality Checking for Complex System Models. In Giacobazzi, R.; Berdine, J.; and Mastroeni, I., eds., *Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20-22, 2013. Proceedings*, volume 7737 of *Lecture Notes in Computer Science*, 248–267. Springer.

Levesque, H. J.; Pirri, F.; and Reiter, R. 1998. Foundations for the Situation Calculus. *Electronic Transactions on Artificial Intelligence (ETAI)*, 2: 159–178.

McCarthy, J.; and Hayes, P. J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4: 463–502.

Pearl, J. 1998. On the Definition of Actual Cause. Technical Report R-259, University of California Los Angeles.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Pirri, F.; and Reiter, R. 1999. Some contributions to the metatheory of the situation calculus. *J. ACM*, 46(3): 325–361.

Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. Cambridge, MA, USA: MIT Press. ISBN 9780262182188.

Rostamigiv, M.; Khan, S. M.; Lespérance, Y.; and Yadkoo, M. 2024. A Logic of Actual Cause for Non-Deterministic Dynamic Domains. In *Working Notes of the 21st European Conference on Multi-Agent Systems, August 26-28th, 2024, Dublin, Ireland*.

Scherl, R. B.; and Levesque, H. J. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1-2): 1–39.

Yazdanpanah, V.; Gerding, E. H.; Stein, S.; Dastani, M.; Jonker, C. M.; Norman, T. J.; and Ramchurn, S. D. 2023. Reasoning about responsibility in autonomous systems: challenges and opportunities. *AI Soc.*, 38(4): 1453–1464.