Knowing Why – On the Dynamics of Knowledge about Actual Causes in the Situation Calculus

Shakil M. Khan
Ronin Institute
Montclair, NJ, USA
shakil.khan@ronininstitute.org

Yves Lespérance
York University
Toronto, Ontario, Canada
lesperan@eecs.yorku.ca

ABSTRACT

Reasoning about observed effects and their causes is important in many applications. For instance, understanding why a plan failed can aid the task of replanning by allowing the agent to tailor a better plan. But under incomplete information, an agent may be unable to determine which actions/events caused an effect. To overcome this, the agent may be able to perform some sensing actions that allow him to figure out what caused the effect. This becomes even more important in multiagent contexts, where an agent may want to identify which agents caused some effect, or possibly prevent other agents from determining who caused something. The effects involved may even be epistemic effects, such as an agent coming to know the PIN of a bank card, and the causes may be sensing actions. Reasoning about such causes is a key part of "theory of mind" and understanding other agents' behaviour. While there has been much work on causality from an objective standpoint, causality from the point of view of individual agents has received much less attention. In this paper, we develop a formalization of knowledge about actual causes in the situation calculus, and how it is affected by actions including sensing. We show that the proposed framework has some intuitive properties and study the conditions under which an agent can be expected to come to know the causes of an effect.

KEYWORDS

Actual Cause; Knowledge; Sensing Actions; Causal Knowledge; Situation Calculus; Logic

ACM Reference Format:
Shakil M. Khan and Yves Lespérance. 2021. Knowing Why – On the Dynamics of Knowledge about Actual Causes in the Situation Calculus. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Actual causality, also known as token-level causality, is a long standing philosophical problem that is intrinsic to the task of reasoning about observations. Given a narrative or trace of events, computing the actual causes of an observed effect involves finding the events in the narrative that are relevant to the effect, i.e. those that caused the effect. This is in contrast to *general* or type-level causality, where the task is to discover universal causal mechanisms. Reasoning about observed effects and their causes is also important for agents. Formalizing knowledge about actual causes in an agent framework can be useful for a variety of tasks. For instance, such reasoning can help an agent to recover from plan failure: information about why a plan failed can aid the task of replanning by allowing the agent to tailor a better plan. But under incomplete information, an agent may be unable to determine which actions/events caused an effect. To overcome this, the agent may be able to perform some sensing actions that allow him to figure out what caused the effect. This becomes even more important in multiagent contexts, where an agent may want to identify which agents caused some effect, or possibly prevent other agents from determining who caused something. The effects involved may even be epistemic effects, such as an agent coming to know the PIN of a bank card, and the causes may be sensing actions. Reasoning about such causes is a key part of "theory of mind" and understanding other agents' behaviour.

Pearl [29, 30] was a pioneer in computational enquiry into actual causality. This line of research was later continued by Halpern and Pearl [13, 16] and others [9, 14, 15, 19, 20]. This "HP approach" is based on the concept of structural equations [35]. HP follows the Humean counterfactual definition of causation, which states that "an outcome B is caused by an event A" is the same as saying that "had A never occurred, B never had existed". This definition suffers from the problem of preemption[1]: it could be the case that in the absence of event A, B would still have occurred due to another event, which in the original trace was preempted by A. HP address this by performing counterfactual analysis only under carefully selected contingencies, which suspend some subset of the model's mechanisms. While their inspirational early work has been shown to be useful for some practical applications (e.g. [4]), their approach based on Structural Equations Models (SEM) has been criticized for its limited expressiveness [11, 19, 20], and researchers have attempted to expand SEM with additional features, e.g. [24].

A different approach was proposed by Batusov and Soutchanski [3], who developed a foundational definition of actual achievement cause within situation calculus basic action theories [31]. They focused on linear traces only. However, an advantage of their approach is that it is based on an expressive formal theory of action.

While there has been much work on actual causality from an objective standpoint, causality from the point of view of some particular agent has received much less attention. In this paper, we develop a formalization of *knowledge about actual causes* in the situation calculus, and how it is affected by actions including sensing. Our formalization, which is based on the definition of actual cause in [3], supports epistemic effects and recognizes sensing actions as causes. We show that the proposed framework has some intuitive properties and study the conditions under which an agent can be expected to come to know the causes of an effect. We also prove

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

[1] Preemption happens when two competing events try to achieve the same effect, and the latter of these fails to do so, as the earlier one has already achieved the effect.

that the regression operator in the situation calculus can be used to answer queries about causal knowledge when the action history is known.

The paper is organized as follows. In the next section, we outline the situation calculus and a model of knowledge, and introduce our running example. In §3, we discuss the definition of actual cause proposed in [3]. Based on this, in §4, we present our logic of actual cause within the situation calculus. In §5, we present our formalization of knowledge about actual causes, discuss how causal knowledge changes as a result of (knowledge-producing) actions, and prove some properties of our formalization. In §6, we show how epistemic causes and effects can be handled. We conclude with some discussion in §7.

2 ACTION AND KNOWLEDGE

Our base framework for modeling causal knowledge is the situation calculus (SC) [27] as formalized in [31]. Here, a possible state of the domain is represented by a situation. The initial state is denoted by S_0. There is a distinguished binary function symbol do where $do(a, s)$ denotes the successor situation to s resulting from performing the action a. Thus the situations can be viewed forming a tree, where the root of the tree is an initial situation and the arcs represent actions. As usual, a relational/functional fluent 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. We will use the abbreviation $do([\alpha_1, \cdots, \alpha_n], S_0)$ to represent the situation obtained by consecutively performing $\alpha_1, \cdots, \alpha_n$ starting from S_0. Also, the notation $s \sqsubset s'$ means that situation s' can be reached from situation s by executing a 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 $s < s' \vee s = s'$.

Our framework uses an action theory $\mathcal{D}$ that includes the following set of axioms:[2] (1) action precondition axioms (APA), one per action a characterizing $Poss(a, s)$, (2) successor state axioms (SSA), one per fluent, that succinctly encode both effect and frame axioms and specify exactly when the fluent changes [31], (3) initial state axioms describing what is true initially, (4) unique name axioms for actions, and (5) domain-independent foundational axioms describing the structure of situations [25, 31].

The SC features a *single-step regression* operator $\rho[\Phi, \alpha]$. Given a query "does Φ hold in the situation obtained by performing the ground action α in some situation σ, i.e. in $do(\alpha, \sigma)$?", ρ transforms it into an *equivalent* query "does Ψ hold in situation σ?", eliminating action α. The expression $\rho[\Phi, \alpha] = \Psi$ denotes such a logically equivalent query obtained from the formula Φ by replacing each fluent atom P in Φ with the right-hand side of the SSA for P where the action variable a is instantiated with the ground action α, and then simplified using unique name axioms for actions and constants. Ψ thus provides the weakest preconditions of Φ in σ given α.

We will use uppercase Greek letters Φ, Ψ, etc. for situation-suppressed SC formulae, which are defined as follows:

$$\Phi ::= P(\vec{x}) \mid \neg\Phi \mid \Phi \wedge \Psi \mid \exists x.\ \Phi,$$

where $\vec{x}$ and x are object terms. Also, we will use α and σ, possibly with decorations, to represent ground action and situation terms, respectively. Finally, we will use uppercase Latin letters for ground terms, and lowercase Latin letters for variables.

Following [28, 32], we model knowledge using a possible worlds account adapted to the SC. There can now be multiple initial situations. $Init(s)$ means that s is an initial situation. The actual initial state is denoted by S_0. $K(s', s)$ is used to denote that in situation s, the agent thinks that she could be in situation s'. Using K, the knowledge of an agent is defined as:[3] $Know(\Phi, s) \stackrel{\text{def}}{=} \forall s'.\ K(s', s) \supset \Phi[s']$, i.e. the agent knows Φ in s if Φ holds in all of her K-accessible situations in s. We also use the abbreviations $KWhether(\Phi, s) \stackrel{\text{def}}{=} Know(\Phi, s) \vee Know(\neg\Phi, s)$, i.e., the agent knows whether Φ holds in s and $KRef(\theta, s) \stackrel{\text{def}}{=} \exists t.\ Know(\theta = t, s)$, i.e., she knows who/what θ refers to. K is constrained to be reflexive and Euclidean (and thus transitive) in the initial situation to capture the fact that the agent's knowledge is true, and that she has positive and negative introspection.

In our framework, the dynamics of knowledge is specified using a SSA for K that supports knowledge expansion as a result of sensing actions. The information provided by a binary sensing action is specified using the predicate $SF(a, s)$. For example, we might have an axiom: $SF(sense_{onTable}(b), s) \equiv onTable(b, s)$, i.e., the action $sense_{ontable}(b)$ will tell the agent whether the block b is on the table in the situation where it is performed. Similarly for non-binary sensing actions, the term $sff(a, s)$ is used to denote the sensing value returned by the action. For example, we might have $sff(read_{numBlocksOnTable}, s) = numBlocksOnTable(s)$, i.e. $read_{numOfBlocksOnTable}$ tells the agent the number of blocks on the table. As shown in [32], the constraints on K then continue to hold after any sequence of actions since they are preserved by the SSA for K. Scherl and Levesque [32] also showed how one can define regression for knowledge-producing actions.

Thus to model knowledge, we will use a theory that is similar to before, but with modified foundational axioms to allow for multiple initial epistemic states. Also, action preconditions can now include knowledge preconditions and initial state axioms can now include axioms describing the epistemic states of the agents. Finally, the aforementioned axioms for K are included. See [31] for details of these. Note that like [32], we assume that actions are fully observable (even if their effects are not). This can be generalized as in [1].

Example. We use a simple blocks-world like domain as our running example. We have an agent/robot that is equipped with a single gripper. The agent is in a room that has at least two different blocks, B_1 and B_2. The agent can pick up (and drop) a block b by executing the $pickUp(b)$ (and $drop(b)$, resp.) action. The agent can only hold one block at a time. Some of the blocks can be fragile. Dropping a fragile block breaks the block. The agent can also make a block b fragile by quenching it, i.e. executing the $quench(b)$ action.[4] Finally, initially the agent is holding block B_1.

[2] We will be quantifying over formulae, and thus assume that $\mathcal{D}$ includes axioms for encoding of formulae as first order terms, as in [34].

[3] Φ can contain a placeholder *now* in the place of the situation terms. Also, $\Phi[s]$ denotes the formula obtained by restoring the situation argument s into all fluents in Φ.

[4] Quenching, which increases the hardness as well as the fragility, involves the rapid cooling of a material to obtain certain properties.

There are three fluents in this domain, $holding(b, s)$, $fragile(b, s)$, and $broken(b, s)$, which respectively mean that the agent is holding block b in situation s, b is fragile in s, and b is broken in s.

We now give the domain-dependent axioms specifying this domain. First, the preconditions for $pickUp(b)$, $drop(b)$, and $quench(b)$ can be specified using APAs as follows (henceforth, all free variables in a sentence are assumed to be universally quantified):

$$(a).\ Poss(pickUp(b), s) \equiv \neg\exists b'.\ holding(b', s),$$
$$(b).\ Poss(drop(b), s) \equiv holding(b, s), \qquad (c).\ Poss(quench(b), s).$$

For instance, (a) says that the agent can pick up a block b in situation s iff she is not already holding another block b'.

Moreover, the following SSAs specify how the fluents *holding*, *fragile*, and *broken* change value when an action happens:

$$(d).\ holding(b, do(a, s)) \equiv (a = pickUp(b) \lor (holding(b, s) \land a \neq drop(b))),$$
$$(e).\ fragile(b, do(a, s)) \equiv (a = quench(b) \lor fragile(b, s)),$$
$$(f).\ broken(b, do(a, s)) \equiv ((fragile(b, s) \land a = drop(b)) \lor broken(b, s)).$$

That is, (d) the agent is holding a block b in the situation resulting from executing action a in situation s (i.e. in $do(a, s)$) if and only if a refers to the agent's action of picking b up from the table, or she already had b in s and a is not the action of dropping b, etc.

Furthermore, the following initial state axioms say that initially (g) the agent is only holding block B_1, (h) all the blocks are non-fragile, and (i) all the blocks are intact:

$$(g).\ \forall b.\ holding(b, S_0) \equiv b = B_1, \qquad (h).\ \forall b.\ \neg fragile(b, S_0),$$
$$(i).\ \forall b.\ \neg broken(b, S_0).$$

Finally, we implicitly assume unique names axioms for blocks, and unique names for actions axioms. Henceforth, we use $\mathcal{D}_{bw}$ to refer to the above axiomatization.

Given this, let us compute the single-step regression $\rho[broken(B_1, do(drop(B_1), s^*)), drop(B_1)]$, for some situation s^*. From the right-hand side of the successor-state axiom (f) above and by substituting action variable a by $drop(B_1)$, object variable b by B_1, and situation variable s by s^*, the result of $\rho[broken(B_1, do(drop(B_1), s^*)), drop(B_1)]$ amounts to $(fragile(B_1, s^*) \land drop(B_1) = drop(B_1)) \lor broken(B_1, s^*)$. Using the unique names axioms, the result of ρ can be simplified to $fragile(B_1, s^*) \lor broken(B_1, s^*)$.

3 ACTUAL CAUSE

Given a trace of events, *actual achievement causes* are the events that are behind achieving an effect while *actual maintenance causes* are those which are responsible for mitigating the threats to the achieved effect.[5] There can also be cases of subtle interactions of these two. In this section, we review previous work on achievement causality in the SC [3]. An effect here is a SC formula $\Phi[s]$ that is *uniform in s* (meaning that it has no occurrences of *Poss*, $\sqsubset$, other situation terms besides s, and quantifiers over situations) and that may include quantifiers over object variables. Given an effect Φ, the actual causes are defined relative to a *causal setting* that includes a theory $\mathcal{D}$ representing the domain dynamics, and a ground situation σ, representing the "narrative" (i.e. trace of events) where the effect was observed.

Definition 3.1 (Causal Setting [3]). A causal setting is a tuple $\langle \mathcal{D}, \sigma, \Phi[s] \rangle$, where $\mathcal{D}$ is a theory, σ is a ground situation term of the form $do([\alpha_1, \cdots, \alpha_n], S_0)$ with ground action functions $\alpha_1, \cdots, \alpha_n$ such that $\mathcal{D} \models executable(\sigma)$, and $\Phi[s]$ is a SC formula uniform in s such that $\mathcal{D} \models \neg\Phi[S_0] \land \Phi[\sigma]$.

As the theory $\mathcal{D}$ does not change, when referring to a causal setting we will often suppress $\mathcal{D}$ and simply write $\langle \sigma, \Phi \rangle$. Also, here Φ is required to hold by the end of the narrative σ, and thus we ignore the cases where Φ is not achieved by the actions in σ, since if this is the case, the achievement cause truly does not exist.

Note that since all changes in the SC result from actions, the potential causes of an effect Φ are identified with a set of ground action terms occurring in σ. However, since σ might include multiple occurrences of the same action, one also needs to identify the situations where these actions were executed.

According to Batusov and Soutchanski [3], if some action α of the action sequence in σ triggers the formula Φ to change its truth value from false to true relative to $\mathcal{D}$, and if there are no actions in σ after α that change the value of Φ back to false, then α is an actual cause of achieving Φ in σ. They showed that when used together with the single-step regression operator ρ, in addition to the single action that brings about the effect of interest, one can also capture the chain of actions that build up to it. The following inductive definition formalizes this intuition. Let $\Pi_{apa}(\alpha, \sigma)$ be the r.h.s. of the precondition axiom for action α in situation σ.

Definition 3.2 (Achievement Cause). A causal setting $C = \langle \sigma, \Phi[s] \rangle$ satisfies the achievement condition of Φ via the situation term $do(\alpha^*, \sigma^*) \sqsubseteq \sigma$ iff there is an action α' and situation σ' such that

$$\mathcal{D} \models \neg\Phi[\sigma'] \land \forall s.\ do(\alpha', \sigma') \sqsubseteq s \sqsubseteq \sigma \supset \Phi[s],$$

and either $\alpha^* = \alpha'$ and $\sigma^* = \sigma'$, or the causal setting $\langle \sigma', \rho[\Phi[s], \alpha'] \land \Pi_{apa}(\alpha', \sigma') \rangle$ satisfies the achievement condition via the situation term $do(\alpha^*, \sigma^*)$. Whenever a causal setting C satisfies the achievement condition via situation $do(\alpha^*, \sigma^*)$, the action α^* executed in situation σ^* is said to be an *achievement cause* in C.

Batusov and Soutchanski [3] show that the achievement causes of C form a finite sequence of situation-action pairs, which they call the *achievement causal chain of* C.

As shown in [2], one can also define the concept of *maintenance cause* by appealing to a counterfactual notion of potential threats in the causal setting that can possibly flip the truth value of the effect Φ to false, and actions in the narrative that mitigated those threats. In general, actual causes can be either achievement causes or maintenance causes and the causal chain can include both. To keep it simple, we focus exclusively on actual achievement causes.

Example (Cont'd). Consider the narrative $\sigma_1 = do([drop(B_1), quench(B_1), quench(B_2), pickUp(B_1), drop(B_1)], S_0)$, i.e. the agent drops the block B_1 (that she is holding), then she quenches the block B_1 and then B_2, then she picks up B_1, and finally she drops it again. We are interested in computing the actual causes of the effect $\Phi_1 = broken(B_1, s)$. This scenario is depicted in the upper part of Figure 1. Here the truth value of fluents $holding(B_1)$, $fragile(B_1)$, and $broken(B_1)$ for each situation can be read by looking at the

[5] We do not conceptually distinguish between agents' actions and nature's events.

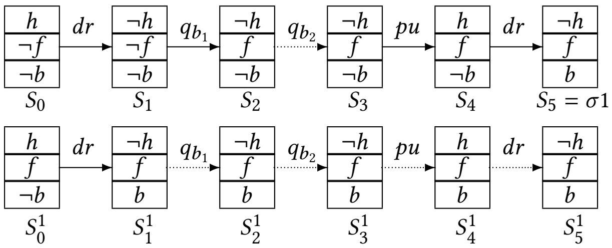

Figure 1: Evolution of fluents starting in situations S_0 and S_0^1

situation box top-down. The actions between each pair of situations are also shown (for now, ignore the second row).

Then, according to Definition 3.2, the causal setting $\langle \sigma_1, \Phi_1 \rangle$ satisfies the achievement condition Φ_1 via the situation term $do(drop(B_1), S_4)$, where $S_4 = do([drop(B_1), quench(B_1), quench(B_2), pickUp(B_1)], S_0)$, so $drop(B_1)$ executed in S_4 is an achievement cause of $broken(B_1)$.

Moreover, let us compute $\rho[broken(B_1, \sigma_1),\ drop(B_1)]$ and $Poss(drop(B_1), S_4)$, starting with the former. As shown in §2 above, the result of ρ can be simplified to $fragile(B_1, S_4) \lor broken(B_1, S_4)$. Let us now consider $Poss(drop(B_1), S_4)$; from the right-hand side of action precondition axiom (b) above and by replacing object variable b with B_1 and situation variable s by S_4, we have $holding(B_1, S_4)$. Computing $\rho[broken(B_1, \sigma_1), drop(B_1)] \land Poss(drop\ (B_1), S_4)$ thus gives rise to a new causal setting $\langle S_4, (fragile(B_1) \lor broken(B_1)) \land holding(B_1)\rangle$. It can be shown that this setting satisfies the achievement condition via the action $pickUp(B_1)$, so $pickUp(B_1)$ executed in $S_3 = do([drop(B_1), quench(B_1), quench(B_2)], S_0)$ is an achievement cause. Furthermore, this yields yet another setting:

$$\langle S_3, \rho[(fragile(B_1, S_4) \lor broken(B_1, S_4)) \land holding(B_1, S_4),\ pickUp(B_1)] \land Poss(pickUp(B_1), S_3)\rangle.$$

Doing simplifications similar to what we did before, we can arrive at the new causal setting $\langle S_3, (fragile(B_1, s) \lor broken(B_1, s)) \land \neg\exists b'.\ holding(b', s)\rangle$, which meets the achievement condition via the action $quench(B_1)$ executed in situation S_1.

And again, this yields another setting:

$$\langle S_1, \rho[(fragile(B_1, S_2) \lor broken(B_1, S_2)) \land \neg\exists b'.\ holding(b', S_2),\ quench(B_1)] \land Poss(quench(B_1), S_1)\rangle,$$

which can be simplified to $\langle \neg\exists b'.\ holding(b', s), S_1\rangle$, and meets the achievement condition via $drop(B_1)$ executed in S_0, and the analysis terminates. The causal chain obtained is thus as follows:

$$\{(drop(B_1), S_4), (pickUp(B_1), S_3), (quench(B_1), S_1), (drop(B_1), S_0)\}.$$

Note that Definition 3.2 can clearly distinguish between irrelevant actions, such as $quench(B_2)$, and actions in the causal chain. The latter are depicted using solid arrows in Figure 1. Also, it can handle *quantified effects*, e.g. $\exists b.\ broken(b, s)$, i.e. the effect that some block was broken.

4 A LOGIC OF ACTUAL CAUSE

While the authors in [3] give a definition of actual cause using the SC, as seen in the previous section their definition is metatheoretic and appeals to regression, a syntactic notion. This makes it hard to use their definition in the context of knowledge. To see the problem, consider the case where our example agent does not know in S_0 whether block B_1 is fragile. Then it can be shown that she does not know in σ_1 what the cause of Φ_1 is. This is because there is a K-alternative situation S_0^1 in S_0 where $fragile(B_1)$ is true and so $broken(B_1)$ is achieved in $do(drop(B_1), S_0^1)$, and it remains true after that (see Figure 1). Hence the causal chain obtained relative to setting $\langle S_5^1, \Phi_1\rangle$ (where $S_5^1 = do([drop(B_1), quench(B_1), quench(B_2), pickUp(B_1), drop(B_1)], S_0^1))$ only includes this action. Moreover, as shown earlier, the causal chain obtained relative to $\langle \sigma_1, \Phi_1\rangle$ is a different one. Thus the agent does not know in σ_1 what the causes of Φ_1 are. However, since in this SC language there is no expression that represents the fact that some action executed in some situation is a cause of some effect, there is no simple way of saying that the agent knows/does not know that this is the case. In other words, if we had a construct $Causes(\dots)$ defined in the language of the SC, then we could have written $Know(Causes(\dots), \sigma_1)$.

Thus, to refer to causal knowledge, we will incorporate such a construct within the language of SC. For this, we will need to generalize the notion of causal settings (see below). We start by introducing the notion of epistemic dynamic formulae in the SC.

Definition 4.1. Let $\vec{x}$, θ_a, and $\vec{y}$ respectively range over object terms, action terms, and object and action terms. The class of *situation-suppressed epistemic dynamic formulae* ψ is defined inductively using the following grammar:

$$\psi ::= P(\vec{x}) \mid Poss(\theta_a) \mid After(\theta_a, \psi) \mid \neg\psi \mid \psi_1 \land \psi_2 \mid \exists\vec{y}.\psi \mid Know(\psi).$$

That is, an epistemic dynamic formula can be a situation-suppressed fluent, a formula that says that some action θ_a is possible, a formula that some epistemic dynamic formula holds after some action has occurred, a formula that can built from other epistemic dynamic formulae using the usual connectives, or a formula that the agent knows that some epistemic dynamic formula holds. Note that ψ can have quantification over object and action variables, but must not include quantification over situations or ordering over situations (i.e. $\sqsubset$). Also, while it may include knowledge modalities, K-relations that do not come from the expansion of *Know* are not permitted. We will use lower-case ψ for epistemic dynamic formulae. If ψ does not include the *Know* modality, we call it a *dynamic formula*.

We use $\psi[s]$ to denote the formula obtained from ψ by restoring the appropriate situation argument into all fluents in ψ. Formally:

Definition 4.2.

$$\psi[s] \stackrel{\text{def}}{=} \begin{cases} P(\vec{x}, s) & \text{if } \psi \text{ is } P(\vec{x}) \\ Poss(\theta_a, s) & \text{if } \psi \text{ is } Poss(\theta_a) \\ \psi'[do(\theta_a, s)] & \text{if } \psi \text{ is } After(\theta_a, \psi') \\ \neg(\psi'[s]) & \text{if } \psi \text{ is } (\neg\psi') \\ \psi_1[s] \land \psi_2[s] & \text{if } \psi \text{ is } (\psi_1 \land \psi_2) \\ \exists\vec{y}.\ (\psi'[s]) & \text{if } \psi \text{ is } (\exists\vec{y}.\ \psi') \\ \forall s'.\ K(s', s) \supset (\psi'[s']) & \text{if } \psi \text{ is } Know(\psi') \end{cases}$$

In the rest of this section, we will use dynamic formulae exclusively. Later in §6, we will come back to epistemic dynamic formulae.

We generalize causal settings by allowing effects in our framework to be any (epistemic) dynamic formula ψ, i.e. we do not require the effect to be uniform in s. Also, we do not require the scenario to be ground and it can include arbitrary (non-ground) action terms. This generalization allows for the seamless incorporation of actual causes within the SC language, especially in the context of knowledge.

Now, since the trace/narrative defined by s (or more precisely, by the situation pair S_0 and s) might include multiple occurrences of the same action, we also need a simple way to identify the situations where these actions were executed. To simplify things, we will require that each situation is associated with a time-stamp. When we move to knowledge, we will have different K-accessible situations where an action occurs, so using time-stamps provides a common reference/rigid designator for the action occurrence. The initial situation S_0 starts at time 0 and each action increments the time-stamp by one. Thus, our theory includes the following axioms:

$$start(S_0) = 0, \qquad \forall a, s, t.\ start(do(a, s)) = t \equiv start(s) = t - 1.$$

With this, we can define a causal chain with respect to a causal setting in our framework as a non-empty set of action-time-stamp pairs derived from the trace s. We don't need to include the situation where the action was executed since the included time-stamp uniquely represents the action's position on the trace.

We are now ready to integrate causes of effects into the SC. We define causes in two steps, starting with primary causes.

Definition 4.3 (Primary Cause).

$$\begin{aligned} &CausesDirectly(a, t, \psi, s) \stackrel{\text{def}}{=} \\ &\qquad \exists s_a.\ start(s_a) = t \wedge (S_0 < do(a, s_a) \leq s) \\ &\qquad \wedge \neg\psi[s_a] \wedge \forall s'.(do(a, s_a) \leq s' \leq s \supset \psi[s']). \end{aligned}$$

That is, an action a executed at time t is the *primary cause* of effect ψ in situation s iff a was executed in a situation with time-stamp t in scenario s, a caused ψ to change its truth value to true, and no subsequent actions on the way to s falsified ψ.

We next generalize this to include indirect causes.[6]

Definition 4.4 (Actual Cause).

$$\begin{aligned} &Causes(a, t, \psi, s) \stackrel{\text{def}}{=} \\ &\forall P.[\forall a, t, s, \psi.(CausesDirectly(a, t, \psi, s) \supset P(a, t, \psi, s)) \wedge \\ &\quad \forall a, t, s, \psi.(\exists a', t', s'.(CausesDirectly(a', t', \psi, s) \wedge start(s') = t' \\ &\qquad\qquad \wedge s' < s \wedge P(a, t, [Poss(a') \wedge After(a', \psi)], s')) \\ &\qquad\qquad\qquad \supset P(a, t, \psi, s)) \\ &\quad] \supset P(a, t, \psi, s). \end{aligned}$$

Thus, *Causes* is defined to be the least relation P such that if a executed at time t directly causes ψ in scenario s then (a, t, ψ, s) is in P, and if a' executed at t' is a direct cause of ψ in s, the time-stamp of s' is t', $s' < s$, and $(a, t, [Poss(a') \wedge After(a', \psi)], s')$ is in P (i.e. a executed at t is a direct or indirect cause of $[Poss(a') \wedge After(a', \psi)]$ in s'), then (a, t, ψ, s) is in P. Here the effect $[Poss(a') \wedge After(a', \psi)]$ requires a' to be executable and ψ to hold after a'.

Note that, the above definitions can handle the trickier case of conditional effects. To see this, consider a simple example, where fluents $fragile(B_2)$ and $broken(B_2)$ are both false initially; as axiomatized, for any situation s, action $quench(B_2)$ executed in s achieves $fragile(B_2)$, and $drop(B_2)$ executed in s achieves $broken(B_2)$, but only when $fragile(B_2)$ holds. As expected, it follows from our definitions that $CausesDirectly(quench(B_2), 0, fragile(B_2), do(quench(B_2), S_0))$ and $CausesDirectly(drop(B_2), 1, broken(B_2), do([quench(B_2), drop(B_2)], S_0))$. Moreover, $quench(B_2)$ executed at 0 can be shown to be the indirect cause of the conditional effect $broken(B_2)$,[7] i.e. $Causes(quench(B_2), 0, broken(B_2), do([quench(B_2), drop(B_2)], S_0))$. This is indeed the case since $Poss(drop(B_2)) \wedge After(drop(B_2), broken(B_2))$ can be shown to hold in $do(quench(B_2), S_0)$, and thus by Definition 4.3, $CausesDirectly(quench(B_2), 0, [Poss(drop(B_2)) \wedge After(drop(B_2), broken(B_2))], do(quench(B_2), S_0))$, and hence by this and Definition 4.4, it follows that $Causes(quench(B_2), 0, broken(B_2), do([quench(B_2), drop(B_2)], S_0))$.

Example (Cont'd). Assume that the axioms for *start* are included in $\mathcal{D}_{bw}$. Then the following lists the causes of Φ_1 in σ_1.

Proposition 4.5 (Causes in σ_1).

$$\begin{aligned} \mathcal{D}_{bw} \models\ & Causes(drop(B_1), 0, \Phi_1, \sigma_1) \wedge Causes(quench(B_1), 1, \Phi_1, \sigma_1) \\ & \wedge Causes(pickUp(B_1), 3, \Phi_1, \sigma_1) \wedge Causes(drop(B_1), 4, \Phi_1, \sigma_1) \\ & \wedge \neg Causes(quench(B_2), 2, \Phi_1, \sigma_1). \end{aligned}$$

On the other hand, if we had modified $\mathcal{D}_{bw}$ to include, instead of Axiom (h), that only B_1 is initially fragile, i.e. $(h').\forall b.fragile(b, S_0) \equiv b = B_1$, then the causes of Φ_1 in σ_1 would have been as follows:

Proposition 4.6.

$$\begin{aligned} &\mathcal{D}_{bw} \setminus \{(h)\} \cup \{(h')\} \models \\ &\qquad \forall a, t.\ Causes(a, t, \Phi_1, \sigma_1) \equiv a = drop(B_1) \wedge t = 0. \end{aligned}$$

5 CAUSAL KNOWLEDGE, SENSING, AND THE DYNAMICS OF CAUSAL KNOWLEDGE

Having defined $Causes(a, t, \psi, s)$, we can now use it just like any other formula in the context of *Know*. We can state that an agent knows in some situation s that a executed at time t is a cause of an effect ψ, i.e. $Know(Causes(a, t, \psi, now), s)$, which by definition of knowledge means that $\forall s'.\ K(s', s) \supset Causes(a, t, \psi, s')$, i.e. in all her epistemic alternatives s', a at t is a cause of ψ.

Returning to our example, assume that the agent initially knows that all the blocks are intact and that she is only holding block B_1:

$$(j).\ Know(\forall b.\ \neg broken(b), S_0),$$
$$(k).\ Know(\forall b.\ holding(b) \equiv b = B_1, S_0).$$

Thus $\neg broken(B_1) \wedge \neg broken(B_2) \wedge holding(B_1) \wedge \neg holding(B_2)$ holds in all of her initial K-accessible worlds/situations. Assume that the agent does not know whether the blocks are fragile:

$$(l).\ \forall b.\ \neg KWhether(fragile(b), S_0).$$

Thus, initially there are at least four possible worlds that are K-related to the initial situation S_0, say S_0, S_0^1, S_0^2, and S_0^3. Each of these worlds assigns a different interpretation to the fragility of the

[6]In this, we need to quantify over situation-suppressed epistemic dynamic formulae. Thus we must encode such formulae as terms and formalize their relationship to the associated situation calculus formulae. This is tedious but can be done essentially along the lines of [10]. We assume that we have such an encoding and use formulae as terms directly.

[7]Recall that $broken(B_2)$'s achievement via $drop(B_2)$ is conditioned on $fragile(B_2)$.

blocks B_1 and B_2. In particular, assume that:

$$(m).\ \forall b.\ \mathit{fragile}(b, S_0^1), \quad (n).\ \forall b.\ \mathit{fragile}(b, S_0^2) \equiv b = B_1,$$
$$(o).\ \forall b.\ \mathit{fragile}(b, S_0^3) \equiv b = B_2.$$

See Figure 1 for S_0 and S_0^1. Now, assume that $\mathcal{D}_{bw}^K$ denotes our axiomatization of the blocks world with knowledge. Then we can show that in situation σ_1, the agent only knows that $drop(B_1)$ executed at time 0 is a cause of Φ_1 and that $quench(B_2)$ executed at 2 is not, but does not know whether other actions on the trace σ_1 are causes:

Proposition 5.1 (Knowledge in σ_1).

$$\begin{aligned}\mathcal{D}_{bw}^K \models\ & Know(Causes(drop(B_1), 0, \Phi_1), \sigma_1) \wedge \\ & Know(\neg Causes(quench(B_2), 2, \Phi_1), \sigma_1) \wedge \\ & \neg KWhether(Causes(quench(B_1), 1, \Phi_1), \sigma_1)) \wedge \\ & \neg KWhether(Causes(pickUp(B_1), 3, \Phi_1), \sigma_1)) \wedge \\ & \neg KWhether(Causes(drop(B_1), 4, \Phi_1), \sigma_1)).\end{aligned}$$

Thus the agent does not know the causal chain: there are common elements, e.g. $drop(B_1)$ executed at time 0, but for other actions such as $drop(B_1)$ executed at 4, the agent is unsure.

However, if the agent were to know in S_0 that B_1 is not fragile, then she would have known the causal chain in σ_1:

Proposition 5.2 (Knowledge in σ_1 (Alternate)).

$$\begin{aligned}&\mathcal{D}_{bw}^K \setminus \{(l)\} \cup \{Know(\neg \mathit{fragile}(B_1), S_0)\} \models \\ &\quad \forall a, t.\ Causes(a, t, \Phi_1, \sigma_1) \equiv Know(Causes(a, t, \Phi_1), \sigma_1).\end{aligned}$$

Now, let us introduce a sensing action $sense_g(b)$ that senses whether block b is made of glass. We introduce a fluent $glass(b, s)$ to state that an object is made of glass. We need to specify its SSA:

$$(p).\ \forall b, a, s.\ glass(b, do(a, s)) \equiv glass(b, s).$$

Also, we need a sensing fluent for $sense_g$; this is specified as follows:

$$(q).\ \forall b, n, s.\ SF(sense_g(b), s) \equiv glass(b, s).$$

That is, $SF(sense_g(b))$ returns the sensing value *true* iff block b is made of glass. Finally, we need an associated initial state axiom, that, initially it is known that a block is fragile iff it is made of glass:

$$(r).\ \forall b.\ Know(\mathit{fragile}(b) \equiv glass(b), S_0).$$

But initially the agent still does not know which blocks are fragile/made of glass. Given this, we can show that despite the incompleteness of her knowledge (about the fragility of B_1) in the initial situation, the agent will learn all the causes of Φ_1 after she senses whether block B_1 is made of glass in σ_1.

Proposition 5.3 (Knowledge in $do(sense_g(B_1), \sigma_1)$ – I).

$$\begin{aligned}\mathcal{D}_{bw}^K \models & Know(Causes(drop(B_1), 0, \Phi_1) \wedge Causes(quench(B_1), 1, \Phi_1) \\ & \quad \wedge Causes(pickUp(B_1), 3, \Phi_1) \wedge Causes(drop(B_1), 4, \Phi_1), \\ & do(sense_g(B_1), \sigma_1)).\end{aligned}$$

To see why this is the case, first note that it follows from $\mathcal{D}_{bw}^K$ that $\neg glass(B_1)$ holds in σ_1. This is because, since by Axiom (h), B_1 was not fragile in S_0, and by Axiom (r), every fragile block in S_0 is made of glass, it follows that B_1 is not made of glass in S_0. Moreover, by Axiom (p) and other axioms in $\mathcal{D}_{BW}^K$, B_1 remains non-glass after any sequence of actions, in particular in σ_1. Furthermore, using similar reasoning it can be shown that the agent knows in σ_1 that any block that is made of glass now was fragile initially and that vice versa. Finally, by this and the SSA for K, in all situations that are K-accessible in $do(sense_g(B_1), \sigma_1)$, $\neg glass(B_1)$ holds, and these are rooted in an initial situation where $\neg \mathit{fragile}(B_1)$ holds. Thus in all her K-alternate worlds in $do(sense_g(B_1), \sigma_1)$, all causes are the same since all of these worlds start with a situation where $\neg \mathit{fragile}(B_1)$ holds.

Moreover, we can show that she will also learn all the non-causes, in particular that $quench(B_2)$ executed at time 2 is not a cause:

Proposition 5.4 (Knowledge in $do(sense_g(B_1), \sigma_1)$ – II).

$$\mathcal{D}_{bw}^K \models Know(\neg Causes(quench(B_2), 2, \Phi_1), do(sense_g(B_1), \sigma_1)).$$

Properties

Let $\mathcal{D}$ be our formalization of causal knowledge. We now show that our formalization has some intuitive properties. First, since the *Causes* operator is defined in the language, one can expect all the properties of knowledge (including knowledge about causes) to follow. Indeed we can show that logically equivalent effects have the same causes and that full introspection holds for causal knowledge.

Next, we identify the conditions under which a (binary) sensing action can be used to learn the causes of an effect.

Theorem 5.5 (From Ignorance to Causal Knowledge).

$$\begin{aligned}\mathcal{D} \models \forall s.\ & executable(s) \wedge \neg\psi[root(s)] \wedge \psi[s] \\ & \wedge \forall s'.\ SF(sense_\Phi, s') \equiv \Phi[s'] \\ & \wedge ((\Phi[s] \wedge \Phi^+(\psi, \Phi, s)) \vee (\neg\Phi[s] \wedge \Phi^-(\psi, \Phi, s)) \\ & \supset \forall a, t.\ KWhether(Causes(a, t, \psi), do(sense_\Phi, s)),\end{aligned}$$

where:

$$\begin{aligned}\Phi^+(\psi, \Phi, s) &\stackrel{\text{def}}{=} (\forall s'.\ K(s', s) \wedge \Phi[s'] \\ &\qquad \supset (\forall a, t.\ Causes(a, t, \psi, s') \equiv Causes(a, t, \psi, s))), \\ \Phi^-(\psi, \Phi, s) &\stackrel{\text{def}}{=} (\forall s'.\ K(s', s) \wedge \neg\Phi[s'] \\ &\qquad \supset (\forall a, t.\ Causes(a, t, \psi, s') \equiv Causes(a, t, \psi, s))),\end{aligned}$$

and,

$$root(s) \stackrel{\text{def}}{=} \begin{cases} root(s') & \text{if } s = do(a', s') \\ s & \text{otherwise.}\end{cases}$$

That is, for any action a and time-stamp t, after performing the binary sensing action $sense_\Phi$ in s, an agent will learn whether a executed at time t is a cause of some effect ψ in scenario s, provided that $sense_\Phi$ senses the value of Φ, and either Φ holds in s and in all the K-accessible worlds s' in s where Φ holds, the causes of ψ in s' are the same as the causes of ψ in s, or Φ does not hold in s and in all the K-accessible worlds s' in s where Φ does not hold, the causes of ψ in s' are the same as those of ψ in s. Thus, when this is the case, the agent can use the sensing action $sense_\Phi$ to learn about all the causes of ψ. Note that, the first three conjuncts $executable(s) \wedge \neg\psi[root(s)] \wedge \psi[s]$ simply guarantee that $\langle s, \psi \rangle$ is a proper causal setting (see Definition 3.2). Theorem 5.5 can be generalized to include non-binary sensing actions and knowledge-producing actions such as an *inform* [21].

We also study the conditions under which action executions do not alter causal knowledge.

Theorem 5.6 (Persistence of Causal Knowledge).

$$\mathcal{D} \models \forall s, s', s^*, a, t.\ executable(s) \wedge \neg\psi[root(s)] \wedge \psi[s] \wedge KWhether(Causes(a, t, \psi), s) \wedge s < s' \wedge (\forall s^*.\ s \leq s^* \leq s' \supset Know(\psi, s^*)) \supset KWhether(Causes(a, t, \psi), s').$$

That is, if an agent knows in s whether an action a executed at time t is a cause of an effect ψ, she will continue to know whether a executed at t is a cause of ψ in a future situation s', provided that her knowledge of the effect ψ does not change between s and s'.

However, this is not the case in general. For instance, if the agent ceases to know that ψ, then in this new situation she will not know what actions are causes. On the other hand, if she knows the causes of ψ in s, knows that ψ became false in some situation after s, but knows that ψ was reachieved later in s', then she may or may not know the causes in s', but some of these causes will certainly be different from what she knew before in s.

Finally, we can show that when the scenario s in the causal setting is ground, our definition of *Causes* expands to a set of first-order formulae, from which causes can be computed via first-order entailment using regression [31] and knowledge regression [32].

Theorem 5.7. *If s is a ground situation term, a is a ground action term, and t is an integer, then $Causes(a, t, \psi, s)$ is equivalent to a regressable formula.*

Proof Sketch. If a executed at time t is a primary cause of ψ in s, it is equivalent to $CausesDirectly(a, t, \psi, s)$, which by Definition 4.3 is regressable when s is ground. On the other hand, if a executed at t is an indirect cause of ψ in s, it can be shown using Definition 4.4 that $Causes(a, t, \psi, s)$ is equivalent to $Causes^{\leq n}(a, t, \psi, s)$ for some n. The latter states that action a executed at t causes ψ in s in a causal chain of at most n steps, and can be expanded to a conjunction of n or less primary cause assertions, i.e. *CausesDirectly* constructs (see [22] for the formal details). Thus this too is regressable. □

Thus while our formulation is second-order, for ground situations computing causes does not require second-order logic.

6 REASONING ABOUT EPISTEMIC CAUSES AND EFFECTS

To enable reasoning about epistemic effects, we allow effects in our framework to be epistemic dynamic formulae, rather than just dynamic formulae. Note that *Know* in such a formula can take an epistemic dynamic formula ψ as argument. As shown in Definition 4.2, $Know(\psi)[s]$ gets expanded to $\forall s'.\ K(s', s) \supset \psi[s']$. We will also allow actions to have knowledge preconditions; *Know* constructs in the context of action preconditions are only allowed to take regular situation-suppressed formulae as argument.

We use a second example to illustrate epistemic causes and effects. We show that as expected, knowledge-producing actions can be causes of epistemic effects, but more interestingly, they can also be causes of physical effects.

In this example, we have a thief T_1 who likes to go to a bank B_1, peek at the customers while they enter their credit card PINs, physically steal the cards, and then withdraw money using the cards and their PINs. There are also actions for waiting in the queue at the bank, leaving the queue, and drinking complimentary coffee at the bank. The actions in this domain can be specified as follows:

$$Poss(goTo(agt, b), s) \equiv \neg at(agt, b, s),$$
$$Poss(waitInQueue(agt, b), s) \equiv at(agt, b, s),$$
$$Poss(leaveQueue(agt, b), s) \equiv waiting(agt, b, s),$$
$$Poss(drinkCoffee(agt, b), s) \equiv at(agt, b, s) \wedge \neg waiting(agt, b, s),$$
$$Poss(steal(agt, cc), s) \equiv \exists agt', b.(has(agt', cc, s) \wedge at(agt', b, s) \wedge at(agt, b, s) \wedge agt' \neq agt),$$
$$Poss(peek_{PIN}(agt, cc), s) \equiv \exists agt', b.(has(agt', cc, s) \wedge at(agt', b, s) \wedge at(agt, b, s) \wedge agt' \neq agt),$$
$$Poss(withdraw(agt, cc, amt), s) \equiv has(agt, cc, s) \wedge KRef(agt, PIN(cc), s).$$

Thus, e.g., an agent *agt* can peek the PIN of a credit card *cc* in some situation s iff *agt* and the agent that currently has *cc* are at the same bank in s. Also, *agt* can withdraw *amt* dollars from credit card *cc* in s iff she has *cc* in s and she knows in s what the PIN of *cc* is.

There is at least one customer A_1 and a credit card CC_1 that A_1 has. The fluents in this domain are $at(agt, b, s)$, $waiting(agt, b, s)$, $has(agt, cc, s)$, $owns(agt, n, s)$, and $PIN(cc, s)$, which mean that agent *agt* is at bank b in situation s, *agt* is waiting in the queue at b in s, *agt* has credit card *cc* in s, *agt* owns n dollars in s, and $PIN(cc, s)$ is the PIN of credit card *cc* in situation s. The successor-state axioms for these are as follows:

$$at(agt, b, do(a, s)) \equiv a = goTo(agt, b) \vee (at(agt, b, s) \wedge \neg\exists b'.\ a = goTo(agt, b')),$$
$$waiting(agt, b, do(a, s)) \equiv a = waitInQueue(agt, b) \vee (waiting(agt, b, s) \wedge \neg a = leaveQueue(agt, b)),$$
$$has(agt, cc, do(a, s)) \equiv a = steal(agt, cc) \vee (has(agt, cc, s) \wedge \neg\exists agt'.(agt' \neq agt \wedge a = steal(agt', cc))),$$
$$owns(agt, amt, do(a, s)) \equiv \exists n, n', cc.\ (owns(agt, n, s) \wedge a = withdraw(agt, cc, n') \wedge amt = n + n') \vee (owns(agt, amt, s) \wedge \neg\exists n, cc.\ a = withdraw(agt, cc, n)),$$
$$PIN(cc, do(a, s)) = p \equiv PIN(cc, s) = p.$$

These are all self-explanatory.

The initial state is specified as follows:[8]

$$Know(agt, owns(T_1, 0), S_0), \quad Know(agt, has(A_1, CC_1), S_0),$$
$$Know(agt, \neg at(T_1, B_1), S_0), \quad Know(agt, at(A_1, B_1), S_0),$$
$$PIN(CC_1, S_0) = 12345, \quad \neg KRef(T_1, PIN(CC_1), S_0).$$

That is, all agents know in the actual initial situation S_0 that agent T_1 has \$0, that agent A_1 has credit card CC_1, that T_1 is not at bank B_1, and that A_1 is at B_1. Also, the PIN of CC_1 in S_0 is 12345 and T_1 does not know what $PIN(CC_1)$ refers to in S_0.

The following sensing-fluent axiom specifies action $peek_{PIN}$:

$$sff(peek_{PIN}(agt, cc), s) = PIN(cc, s).$$

Thus, $peek_{PIN}(agt, cc)$ tells *agt* the PIN of the *cc* in s. As per the SSA for K, other agents see that the $peek_{PIN}$ action has occurred, but do not learn the PIN.

Finally, we assume that other necessary axioms such as unique-names for actions axioms are also specified.

Now, consider the following scenario: $\sigma_2 = do([goTo(T_1, B_1), waitInQueue(T_1, B_1), peek_{PIN}(T_1, CC_1), leaveQueue(T_1, B_1), steal(T_1,$

[8] Following [33], here we use an agent argument in *Know*.

CC_1), $drinkCoffee(T_1, B_1)$, $withdraw(T_1, CC_1, 500)], S_0)$, i.e. T_1 goes to bank B_1, waits in the queue there, peeks at the PIN of credit card CC_1, leaves the queue, steals CC_1, drinks coffee, and then withdraws \$500 from CC_1. We are interested in computing the causes of both $\psi^1_{bank} = KRef(T_1, PIN(CC_1))$ and $\psi^2_{bank} = owns(T_1, 500)$.

Let $\mathcal{D}^K_{bank}$ be our example theory. Then we can show that T_1 knows in σ_2 that the causes of her knowing the PIN of CC_1 are her actions of going to the bank executed at time 0 and peeking at the PIN of CC_1 executed at time 2:

Proposition 6.1.

$$\mathcal{D}^K_{bank} \models \forall a, t.\ Know(T_1, Causes(a, t, \psi^1_{bank}), \sigma_2) \equiv$$
$$(a = goTo(T_1, B_1) \wedge t = 0) \vee (a = peek_{PIN}(T_1, CC_1) \wedge t = 2).$$

As expected, our epistemic effect ψ^1_{bank} is caused by the knowledge-producing action $peek_{PIN}(T_1, CC_1)$ executed at time 2. Note that, the SSA for K drops from the set of K-accessible situations in $do(peek_{PIN}(T_1, CC_1), S_0)$ the situations where the sensed fluent function $sff(peek_{PIN}(T_1, CC_1))$ has a different value from that in the actual situation, and thus afterwards the agent gets to know the value of the PIN, and that of the sensed fluent. Also, since $peek_{PIN}(T_1, CC_1)$ was executable only when T_1 is in the bank B_1, which is brought about by $goTo(T_1, B_1)$ executed at time 0, it is also a cause of ψ^1_{bank} in σ_2.

More interestingly, we can show that T_1 knows in σ_2 that the causes of her having \$500 include, among other actions, the sensing action of her peeking at the PIN of CC_1 executed at time 2:

Proposition 6.2.

$$\mathcal{D}^K_{bank} \models \forall a, t.\ Know(T_1, Causes(a, t, \psi^2_{bank}), \sigma_2) \equiv$$
$$(a = goTo(T_1, B_1) \wedge t = 0) \vee (a = peek_{PIN}(T_1, CC_1) \wedge t = 2) \vee$$
$$(a = steal(T_1, CC_1) \wedge t = 4) \vee (a = withdraw(T_1, CC_1, 500) \wedge t = 6).$$

Thus, interestingly, our framework allows causes of physical effects to be knowledge-producing actions. In particular, this happens when physical actions have knowledge preconditions. For instance, in the above example, one of the preconditions of withdrawing money from a credit card cc is to know what $PIN(cc)$ refers to.

When the action history is ground, it can be shown that one can use knowledge regression [32] to compute causal knowledge, e.g., $Know(Causes(goTo(T_1, B_1), 0, \psi^1_{bank}), \sigma_2)$ above; see [22] for details.

7 DISCUSSION

Based on a formal notion of causality [3], in this paper we developed a logic of actual causes and proposed an account of causal knowledge in the SC. Our account allows agents to have incomplete initial knowledge and can deal with epistemic causes and effects. We showed that it is possible to have different causes of the same effect in different epistemic alternatives. Thus, as expected, 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.

While analyzing actual causes is quite subtle and involves tricky cases including preemption or overdetermination, we did not discuss these here. For instance, we could have argued that in our first example, $drop(B_1)$ executed at time 4 is not a cause of Φ_1 in S^1_5 since it was preempted by the earlier $drop(B_1)$ action executed at time 0: the second *drop* action would have been a cause had the first one not brought about the effect. Nonetheless, we emphasize that our focus here was on studying the epistemics of causality by embedding an existing definition of causation [3] in the SC language rather than proposing a new one. It has been shown that the definition of actual cause in [3] handles –within its limitations– all the paradigmatic examples of causation correctly; see [2, 3] for details. Moreover, Khan and Soutchanski [23] recently showed that their own definition of actual cause derived from a counterfactual standpoint is equivalent to this definition. They also related their definition to a regularity account of causation [26], thus integrating two highly influential but opposing approaches to causation. Also, the counterfactuals they studied preserve the underlying causal relations, which is desirable; this is in contrast to many SEM-based definitions [15], where the counterfactuals are "explicitly nonforetracking" [18].[9]

Our account of causal knowledge relies on key features of the SC and basic action theories, such as SSAs. This also allows the use of knowledge regression to compute causes when the action history is ground. To move to a different logical framework one would need similar machinery as in the SC. For how this can be provided in Dynamic Epistemic Logic, see [36].

Recently, there has been some work that formalizes causality in an epistemic context. For example, while defining responsibility/blame in legal cases, Chockler et al. [8] modeled an agent's uncertainty of the causal setting using an "epistemic state", which is a pair (K, Pr), where K is a set of causal settings and Pr is a probability distribution over K. Their model is based on structural equations. We on the other hand study the epistemics of causality based on the more expressive first-order formalism proposed by Batusov and Soutchanski [3]. Moreover, unlike [8], our account incorporates a formal model of domain dynamics and knowledge change. This allows for an interesting interplay between causality and knowledge. For instance, in our framework it is possible to specify a domain where the agent does not know the causes of an effect in some situation, but learns them after performing some sensing action. To the best of our knowledge, ours is the only formal account of actual causality that investigates the dynamics of causal knowledge and allows knowledge-producing actions to be causes. Among other related work, let us mention logics of "sees to it that" (STIT), which attempt to capture intentional choice/responsibility of agents [6, 17]. Here we are more interested in capturing the causal chain of actions that led to an effect rather than attributing responsibility to agents. Some of our future work include defining responsibility and blame using our formalization.

Here, we assumed that all actions were fully observable; incorporating partial observability of actions as in [1] would yield a more expressive framework. Moreover, here we focused on knowledge and not belief. The relation between actual causes and causal knowledge becomes more intricate if we incorporate the latter. Also, here we focus on deterministic actions only. However, there are several proposals on how one can reason about non-deterministic/stochastic actions in the SC, e.g. [1, 5, 7]. Dealing with these is future work. Finally, in the future we would like to investigate reasoning that involves multiple agents and examine how regression can be employed to evaluate complex causal knowledge queries.

[9] Hall [12] pointed out that the analysis of actual causation using non-actual worlds where the causal relations themselves do not hold is counterintuitive.

REFERENCES

[1] Fahiem Bacchus, Joseph Y. Halpern, and Hector J. Levesque. 1999. Reasoning about Noisy Sensors and Effectors in the Situation Calculus. *Artificial Intelligence* 111, 1-2 (1999), 171–208. https://doi.org/10.1016/S0004-3702(99)00031-4

[2] Vitaliy Batusov and Mikhail Soutchanski. 2017. Situation Calculus Semantics for Actual Causality. In *Proceedings of the Thirteenth International Symposium on Commonsense Reasoning, COMMONSENSE 2017, London, UK, November 6-8, 2017 (CEUR Workshop Proceedings, Vol. 2052)*, Andrew S. Gordon, Rob Miller, and György Turán (Eds.). CEUR-WS.org. http://ceur-ws.org/Vol-2052/paper2.pdf

[3] Vitaliy Batusov and Mikhail Soutchanski. 2018. Situation Calculus Semantics for Actual Causality. In *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*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 1744–1752. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16891

[4] Ilan Beer, Shoham Ben-David, Hana Chockler, Avigail Orni, and Richard J. Trefler. 2012. Explaining Counterexamples using Causality. *Formal Methods in System Design* 40, 1 (2012), 20–40. https://doi.org/10.1007/s10703-011-0132-2

[5] Vaishak Belle and Hector J. Levesque. 2018. Reasoning about Discrete and Continuous Noisy Sensors and Effectors in Dynamical Systems. *Artificial Intelligence* 262 (2018), 189–221. https://doi.org/10.1016/j.artint.2018.06.003

[6] Nuel Belnap, Michel Perloff, and Ming Xu. 2001. *Facing the Future: Agents and Choices in our Indeterministic World.* Oxford University Press.

[7] Craig Boutilier, Raymond Reiter, and Bob Price. 2001. Symbolic Dynamic Programming for First-Order MDPs. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, Bernhard Nebel (Ed.). Morgan Kaufmann, 690–700.

[8] Hana Chockler, Norman E. Fenton, Jeroen Keppens, and David A. Lagnado. 2015. Causal Analysis for Attributing Responsibility in Legal Cases. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law, ICAIL 2015, San Diego, CA, USA, June 8-12, 2015*, Ted Sichelman and Katie Atkinson (Eds.). ACM, 33–42. https://doi.org/10.1145/2746090.2746102

[9] Thomas Eiter and Thomas Lukasiewicz. 2002. Complexity Results for Structure-based Causality. *Artificial Intelligence* 142, 1 (2002), 53–89. https://doi.org/10.1016/S0004-3702(02)00271-0

[10] Giuseppe De Giacomo, Yves Lespérance, and Hector J. Levesque. 2000. ConGolog, A Concurrent Programming Language based on the Situation Calculus. *Artificial Intelligence* 121, 1-2 (2000), 109–169. https://doi.org/10.1016/S0004-3702(00)00031-X

[11] Clark Glymour, David Danks, Bruce Glymour, Frederick Eberhardt, Joseph D. Ramsey, Richard Scheines, Peter Spirtes, Choh Man Teng, and Jiji Zhang. 2010. Actual Causation: A Stone Soup Essay. *Synthese* 175, 2 (2010), 169–192. https://doi.org/10.1007/s11229-009-9497-9

[12] Ned Hall. 2007. Structural Equations and Causation. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition* 132, 1 (2007), 109–136.

[13] Joseph Y. Halpern. 2000. Axiomatizing Causal Reasoning. *Journal of Artificial Intelligence Research* 12 (2000), 317–337. https://doi.org/10.1613/jair.648

[14] Joseph Y. Halpern. 2015. A Modification of the Halpern-Pearl Definition of Causality. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, Qiang Yang and Michael J. Wooldridge (Eds.). AAAI Press, 3022–3033. http://ijcai.org/Abstract/15/427

[15] Joseph Y. Halpern. 2016. *Actual Causality.* MIT Press.

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

[17] Andreas Herzig, Emiliano Lorini, and Nicolas Troquard. 2018. Action Theories. In *Introduction to Formal Philosophy*, Sven Ove Hansson and Vincent F. Hendricks (Eds.). Springer International Publishing, Cham, 591–607.

[18] Christopher Hitchcock. 2001. The Intransitivity of Causation Revealed in Equations and Graphs. *The Journal of Philosophy* 98, 6 (2001), 273–299.

[19] Mark Hopkins. 2005. *The Actual Cause: From Intuition to Automation.* Ph.D. Dissertation. University of California Los Angeles.

[20] Mark Hopkins and Judea Pearl. 2007. Causality and Counterfactuals in the Situation Calculus. *Journal of Logic and Computation* 17, 5 (2007), 939–953. https://doi.org/10.1093/logcom/exm048

[21] Shakil M. Khan and Yves Lespérance. 2005. ECASL: A Model of Rational Agency for Communicating Agents. In *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, Frank Dignum, Virginia Dignum, Sven Koenig, Sarit Kraus, Munindar P. Singh, and Michael J. Wooldridge (Eds.). ACM, 762–769. https://doi.org/10.1145/1082473.1082590

[22] Shakil M. Khan and Yves Lespérance. 2021. *Causal Knowledge: Semantics and Dynamics.* Technical Report. Department of Electrical Engineering and Computer Science, York University, Toronto, Canada.

[23] Shakil M. Khan and Mikhail Soutchanski. 2020. Necessary and Sufficient Conditions for Actual Root Causes. In *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)*, Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang (Eds.). IOS Press, 800–808. https://doi.org/10.3233/FAIA200169

[24] Florian Leitner-Fischer and Stefan Leue. 2013. Causality Checking for Complex System Models. In *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)*, Roberto Giacobazzi, Josh Berdine, and Isabella Mastroeni (Eds.). Springer, 248–267. https://doi.org/10.1007/978-3-642-35873-9_16

[25] Hector J. Levesque, Fiora Pirri, and Raymond Reiter. 1998. Foundations for the Situation Calculus. *Electronic Transactions on Artificial Intelligence (ETAI)* 2 (1998), 159–178. http://www.ep.liu.se/ej/etai/1998/005/

[26] John Leslie Mackie. 1965. Causes and Conditions. *American Philosophical Quarterly* 2, 4 (1965), 245–264.

[27] John McCarthy and Patrick J. Hayes. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence* 4 (1969), 463–502.

[28] Robert C. Moore. 1985. A Formal Theory of Knowledge and Action. In *Formal Theories of the Commonsense World.* Ablex, 319–358.

[29] Judea Pearl. 1998. *On the Definition of Actual Cause.* Technical Report R-259. University of California Los Angeles.

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

[31] Raymond Reiter. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems.* MIT Press, Cambridge, MA, USA.

[32] Richard B. Scherl and Hector J. Levesque. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144, 1-2 (2003), 1–39. https://doi.org/10.1016/S0004-3702(02)00365-X

[33] Steven Shapiro, Yves Lespérance, and Hector J. Levesque. 2002. The Cognitive Agents Specification Language and Verification Environment for multiagent systems. In *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings.* ACM, 19–26. https://doi.org/10.1145/544741.544746

[34] Steven Shapiro, Yves Lespérance, and Hector J. Levesque. 2007. Goal Change in the Situation Calculus. *Journal of Logic and Computation* 17, 5 (2007), 983–1018. https://doi.org/10.1093/logcom/exm050

[35] Herbert A. Simon. 1977. Causal Ordering and Identifiability. *Models of Discovery. Boston Studies in the Philosophy of Science* 54 (1977).

[36] Hans P. van Ditmarsch, Andreas Herzig, and Tiago de Lima. 2007. Optimal Regression for Reasoning about Knowledge and Actions. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada.* AAAI Press, 1070–1076. http://www.aaai.org/Library/AAAI/2007/aaai07-170.php