On Explaining Agent Behaviour via Root Cause Analysis: A Formal Account Grounded in Theory of Mind

Shakil M. Khan a;* and Maryam Rostamigiv a

aUniversity of Regina, Saskatchewan, Canada

Abstract. Inspired by a novel action-theoretic formalization of actual cause, Khan and Lespérance (2021) recently proposed a first account of causal knowledge that supports epistemic effects, models causal knowledge dynamics, and allows sensing actions to be causes of observed effects. To date, no other study has looked specifically at these issues. But their formalization is not sufficiently expressive to model explanations via causal analysis of mental states as it ignores a crucial aspect of theory of mind, namely motivations. In this paper, we build on their work to support causal reasoning about conative effects. In our framework, one can reason about causes of motivational states, and we allow motivation-altering actions to be causes of observed effects. We illustrate that this formalization along with a model of goal recognition can be utilized to explain agent behaviour in communicative multiagent contexts.

1 Introduction

Actual causality is a long-standing philosophical problem that is fundamental to the task of reasoning about and analysing observations. Given a narrative or history of events and an observed effect, solving this problem involves finding the events or actions from this history that are responsible for producing this effect, i.e. those that caused the effect. Also known as token-level causality, this problem is different from general or type-level causality, where the task is to discover universal causal mechanisms. Actual causality plays a significant role in reasoning about agents. For instance, causal reasoning can be used to explain the behaviour of a group of agents, e.g. via causal analysis of the mental states produced by this behaviour. These mental states may include beliefs and goals of the agents whose actions are the cause of the observed behaviour as well as those of others.

Pearl [31, 32] was a pioneer in computational enquiry into actual causality. This line of research was later continued by Halpern [11], Halpern and Pearl [14], and others [7, 16, 17, 12, 13]. This “HP approach” is based on the concept of structural equations [40]. 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: 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 was shown to be useful for some practical applications, their approach based on Structural Equations Models (SEM) has been criticized for its limited expressiveness [16, 17, 10], and researchers have attempted to expand SEM with additional features, e.g. [23]. Note that despite recently reported progresses (e.g. [15]), many of these expressive limitations remain. Also, while there has been much work on actual causality, the vast majority of the work in this area has focused on defining causes from an objective standpoint.

In recent years, researchers have become increasingly interested in studying causation from the perspective of agents. Among other things, this is useful for defining important concepts such as responsibility and blame. Inspired by a novel action-theoretic formalization of actual causation [3], Khan and Lespérance [22] (KL, henceforth) recently proposed a first account of causal knowledge that supports epistemic effects, models causal knowledge dynamics, and allows sensing actions to be causes of observed effects. To date, no other study has looked specifically at these issues. But their formalization is not sufficiently expressive to model explanations via causal analysis of mental states as it ignores a crucial aspect of theory of mind, namely motivations. In this paper, we build on their work to support causal reasoning about conative effects. In our framework, one can reason about causes of motivational states, and we allow motivation-altering actions to be causes of observed effects. We illustrate that this formalization along with a model of goal recognition can be utilized to explain agent behaviour.

Our contribution in this paper is three-fold. First, we show how causal reasoning about goals/intentions can be modeled. Secondly, using an example, we illustrate how this formalization along with a model of goal recognition can be used to explain agent behaviour in communicative multiagent contexts. The generated explanations include both direct causal explanations as well as higher-order and more useful indirect explanations. The latter is grounded in (multiagent) theory of mind-based causal reasoning. Specifically, for this we define explanations as causes of intentions behind other explanations. Finally, while doing this, we extend a previously proposed account of goal change to deal with the request communicative action.

The paper is organized as follows. In the next section, we outline the situation calculus, a model of knowledge therein, and the formalization of infinite paths in the situation calculus, and introduce our running example. In Section 3, we discuss the formalization of prioritized goals and intentions, and propose a modification to the goal dynamics in [19] to deal with requests. Based on this, in Section 4, we present our logic of actual cause within the situation calculus that

* Corresponding Author. Email: shakil.khan@uregina.ca.
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.
can deal with conative effects. In Section 5, we illustrate how our formalization can be utilized to explain agent behaviour. In Section 6, we prove some intuitive properties of our formalization. We conclude with some discussion in Section 7.

2 Action and Knowledge

The Situation Calculus Our base framework for modeling causal reasoning is the situation calculus (SC) [28] as formalized in [34]. 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 $do([α_1, \ldots, α_n], S_0)$ to represent the situation obtained by consecutively performing $α_1, \ldots, α_n$ starting from $S_0$. Also, the notation $s ⊑ s'$ means that situation $s'$ can be reached from situation $s$ by executing a sequence of actions. $s ⊑ s'$ is an abbreviation of $s \sqcup s' \vee s = s'$, $s < s'$ is an abbreviation of $s \sqcup s' \wedge \text{executable}(s')$, where executable($s$) is defined as $∀α'(s, s') do(α', s') ⊑ s ⊃ Poss(α', s')$, 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 $D$ that includes the following set of axioms:\footnote{We will be quantifying over formulae, and thus assume that $D$ includes axioms for encoding of formulae as first order terms, as in [38].} (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 changes, (3) initial state axioms describing what is true initially, (4) unique names axioms for $forall$ actions, and (5) domain-independent foundational axioms describing the structure of situations [26].

Knowledge in the Situation Calculus Following [30, 35], 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 $S_0$. $K(d, s', s)$ is used to denote that in situation $s$, the agent $d$ thinks that it could be in situation $s'$. Using $K$, the knowledge of an agent $d$ is defined as:\footnote{The state formula $Φ$ can contain a placeholder $now$ in the place of the situation terms. We often suppress $now$ when the intent is clear from the context. Also, $Φ(s)$ denotes the formula obtained by restoring the situation argument $s$ into all fluents in $Φ$.} $Know(d, Φ, s) ≡ ∀s'. K(d, s', s) ⊃ Φ(s')$, i.e. the agent $d$ knows $Φ$ in $s$ if $Φ$ holds in all of its $K$-accessible situations in $s$. We also use the abbreviations $Kωether(d, Φ, s) ≡ Know(d, Φ, s) \lor Know(d, \neg Φ, s)$, i.e. $d$ knows whether $Φ$ holds in $s$ and $Kref(d, θ, s) ≡ \exists t. K(d, θ = t, s)$, i.e. it knows who/what $θ$ refers to in $s$. $K$ 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 it 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 as well as communication actions. The information provided by a binary sensing action is specified using the predicate $SF(a, s)$. Similarly for non-binary sensing actions, the term $sf(a, s)$ is used to denote the sensing value returned by the action. These are specified using sensed fluent axioms; see [25] for details. Shapiro et al. [37] and later Lespérance [24] extended the SSA for $K$ to support variants of the ‘inform’ communicative action. We will adopt the variant proposed in KL [18]. The preconditions of $inform(\inf, \agt, Φ)$, which can be used by $\inf$ to inform $\agt$ that $Φ$, are as follows:

$$Poss(inform(\inf, \agt, Φ), s) \equiv Know(\inf, \Phi, s) \land \neg Know(inform(\inf, Know(\agt, Φ, now)), s).$$

We assume that its effects have been specified as in KL [18]. As shown in [35], the constraints on $K$ then continue to hold after any sequence of actions since they are preserved by the SSA for $K$. A similar result can be shown for the KL [18] variant of the SSA for $K$.

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 $K$ and $inform$ are included. See [34] and [18] for details of these. Note that like [35], we assume that actions are fully observable (even if their effects are not). This can be generalized as in [1].

Paths in the Situation Calculus Following KL [21], we will formalize the sort of paths in the SC. A path is essentially an infinite sequence of situations, where each situation along the path can be reached by performing some executable action in the preceding situation. We will use $Starts(p, s)$ to denote that $s$ is the earliest situation on path $p$ and $OnPath(p, s)$ to denote that $s$ is on $p$. $Suffix(p', p, s)$ means that path $p'$ that starts with situation $s$ is a suffix of $p$. KL [20] showed how one can interpret arbitrary CTL\(^*\) formulae within SC with paths.\footnote{CTL\(^*\) [8] is a well-known branching-time temporal logic. It is a superset of Linear Temporal Logic (LTL) [33] and Computational Tree Logic (CTL) [5], and it allows arbitrary mixing of temporal operators and path quantifiers.} We assume that our theory $D$ includes the axiomatization for paths.

We will use uppercase and lowercase Greek letters for state formulae (i.e. situation-suppressed SC formulae) and path formulae, respectively. These are inductively defined as follows:

$$Φ ::= P(\vec{x}) \mid AΦ \mid Φ \land Φ \mid \neg Φ \mid ∀x. Φ \mid Φ$$

$$\phi ::= Φ \mid Φ \land Φ \mid \neg Φ \mid ∀x. Φ \mid ∃x. Φ \mid \phi U \phi$$

Here, $\vec{x}$ and $x$ are object terms, $P(\vec{x})$ is an arbitrary situation-suppressed SCC formula, and $A\phi$ (i.e. over all paths $\phi$) is a path quantifier. Also, $∃\phi$ means that $\phi$ holds next over a path while $ϕ U ψ$ stands for $ϕ until ψ$. Other logical connectives and quantifiers such as $\lor, \land, \equiv, \exists$ and CTL\(^*\) operators such as $Eϕ, ∃ϕ, ϕ ^ B ψ$, etc. are handled as the usual abbreviations (see below).

Like now in state formulae, path formulae $ϕ$ can also contain an often-suppressed path placeholder $path$ in the place of the path terms. The function $[\cdot]$ translates the above-defined formulae into formulae of the SC with paths. We write $Φ[\cdot] (and \phi[\cdot]) to mean that state formula $Φ$ (and path formula $ϕ$) holds in situation $s$ (and over path $p$, respectively). In the following, we give the definition of
action mentioned above, there are three additional actions in this domain. Action takeOff\((d, l)\) can be used by drone \(d\) to take off from location \(l\), flyTo\((d, l, l')\) takes \(d\) from \(l\) to \(l'\), and land\((d, l)\) makes \(d\) land at \(l\). There are four fluents in this domain, \(At(d, l, s)\), \(Flying(d, l)\), \(Vis(d, l, s)\), and \(TStorm(l, s)\), representing that \(d\) is located at \(l\) in situation \(s\), that \(d\) is flying in \(s\), that \(d\) has visited \(l\) in \(s\), and that there is an ongoing thunderstorm at \(l\) in \(s\).

The action preconditions in this domain are as follows:

\[(b). \ \text{Poss}(\text{takeOff}(d, l), s) \equiv At(d, l, s) \land \neg \text{Flying}(d, s),\]
\[(c). \ \text{Poss}(\text{flyTo}(d, l, l'), s) \equiv At(d, l, s) \land \text{Flying}(d, s) \land \text{Route}(l, l') \land \neg \text{Know}(d, TStorm(l'), s),\]
\[(d). \ \text{Poss}(\text{land}(d, l), s) \equiv At(d, l, s) \land \text{Flying}(d, s).\]

Thus, e.g., (c) states that a drone agent \(d\) can fly from locations \(l\) to \(l'\) in situation \(s\) iff it is located at \(l\) in \(s\), it is flying in \(s\), there is a route from \(l\) to \(l'\), and it does not know that there is a storm at \(l'\) in \(s\).

Moreover, the SSA for the above fluents are as follows.

\[(e). \ \text{At}(d, l, \text{do}(a, s)) \equiv \exists l', a = \text{flyTo}(d, l, l') \vee (\text{At}(d, l, s) \land \neg \exists l'. a = \text{flyTo}(d, l, l'))],
\[(f). \ \text{Flying}(d, \text{do}(a, s)) \equiv \exists l. a = \text{takeOff}(d, l) \vee (\text{Flying}(d, s) \land \neg \exists l. a = \text{land}(d, l))],
\[(g). \ \text{Vis}(d, l, \text{do}(a, s)) \equiv \exists l'. a = \text{flyTo}(d, l', l) \vee \text{Vis}(d, l, s),\]
\[(h). \ \text{TStorm}(l, \text{do}(a, s)) \equiv \text{TStorm}(l, s).\]

Thus, e.g., Axiom (e) states that \(d\) is at location \(l\) after executing action \(a\) in situation \(s\) (i.e. in \(\text{do}(a, s)\)) iff \(a\) refers to \(d\)’s action of flying from some location \(l'\) to \(l\), or \(d\) was already at \(l\) in \(s\) and \(a\) is not its action of flying to a different location \(l'\). Note that, for simplicity, we essentially treat \(\text{TStorm}(l, s)\) as a non-fluent.

Initially, drone \(D_1\) is at location \(L_1\), is not flying, and has only visited \(L_1\), and it knows these facts. Moreover, it does not know that there is a storm at location \(L_1\), but knows that there are no storms at \(L_1'\) and \(L_4\). There is indeed a thunderstorm at location \(L_1\) and the controller agent \(D_2\) knows this. Finally, \(D_2\) does not know however that the other agents know this fact. These are captured using the following initial state axioms (note that \(\text{Know}(d, \Phi(\text{now}, s)) \supset \Phi(s)\)):

\[(i). \ \text{Know}(D_1, \text{At}(D_1, L_1), S_0),
(j). \ \text{Know}(D_1, \neg \text{Flying}(D_1), S_0),
(k). \ \text{Know}(D_1, \forall l. \text{Vis}(D_1, l) \equiv l = L_1, S_0),
(l). \ \neg \text{Know}(D_1, \text{TStorm}(L_1), S_0),
(m). \ \text{Know}(D_1, \neg \text{TStorm}(L_1'), S_0),
(n). \ \text{Know}(D_1, \neg \text{TStorm}(L_4), S_0),
(o). \ \text{Know}(D_2, \text{TStorm}(L_1), S_0),
(p). \ \forall d. d \neq D_2 \supset \neg \text{Know}(D_2, \text{Know}(d, \text{TStorm}(L_1)), S_0).\]

3 Formalizing Goals and Intentions

To model conative effects in the SC, we adopt the expressive formalization of prioritized goals (p-goals) and intentions proposed by KL [19]. In this framework, each p-goal is specified by its own accessibility relation \(G\). To deal with multiple agents, we modify KL’s proposal by adding an agent argument for all goal-related predicates and relations; usually the first argument for this. Given agent
d, a path p is $G$-accessible at priority level n in situation s, denoted by $G(d, p, n, s)$, iff the goal of d at level n is satisfied over p and p starts with a situation that has the same action history as s. The latter requirement ensures that the agent’s p-goal-accessible paths reflect the actions that have been performed so far. A smaller n represents higher priority, with 0 being the highest priority level. Thus the set of p-goals are totally ordered according to priority.

We say that d has the p-goal that $\phi$ at level n in situation s iff $\phi$ holds over all paths that are $G$-accessible for d at n in s, i.e. $PGoal(d, \phi, n, s) \equiv \forall p. G(d, p, n, s) \supset \phi[p]$.

We assume that a domain theory $D$ for our framework also includes the domain-dependent initial goal axioms (see below) and the domain-independent axioms and definitions that appear throughout this paper. As KL, we allow the agent to have infinitely many goals, some of which can be left unspecified.

For instance, assume that initially, our drone agent $D_1$ has the following two p-goals: $\phi_0 = \Diamond \text{At}(D_1, \text{L}_d)$, i.e. that it is eventually at $L_d$, and $\phi_1 = \text{Vis}(D_1, \text{L}_d) \lor \text{Vis}(D_1, \text{L}_s)$, i.e. that it visits $L_1$ before it visits $L_2$, at level 0 and 1, respectively. Also, $D_1$ does not have any initial p-goals. Then the initial goal hierarchy of $D_1$ and $D_n$ can be specified using the following axioms:

$q) \quad \text{Init}(s) \supset ((G(D_1, p, 0, s) \equiv \exists s'. \text{Starts}(p, s') \land \text{Init}(s') \land \phi_0[p]))$

$\land (G(D_1, p, 1, s) \equiv \exists s'. \text{Starts}(p, s') \land \text{Init}(s') \land \phi_1[p]))$.

$r) \quad \text{Init}(s) \land n \geq 2 \supset (G(D_1, p, n, s) \equiv \exists s'. \text{Starts}(p, s') \land \text{Init}(s'))$,

$s) \quad \text{Init}(s) \land n \geq 2 \supset (G(D_n, p, n, s) \equiv \exists s'. \text{Starts}(p, s') \land \text{Init}(s'))$.

$q$ specifies the p-goals $\phi_0, \phi_1$ (from lowest to highest priority) of $D_1$ in the initial situations, and makes $G(D_1, p, n, s)$ true for every path p that starts with an initial situation and over which $\phi_0$ holds, for $n = 0, 1$; each of them defines a set of initial goal paths for a given priority level, and must be consistent. (r) makes $G(D_1, p, n, s)$ true for every path p that starts with an initial situation for $n \geq 2$. Thus at levels $n \geq 2$, $D_1$ has the trivial p-goal that it be in an initial situation. The case for $D_n$ is similar.

Assume that $D_{dr}$ denotes our theory for the drone domain. Then in our example, we can show the following:

Proposition 1.

For $n < 2$.

$D_{dr} \models PGoal(D_1, \phi_0 \land \text{Starts}(s) \land \text{Init}(s), n, S_0)$.

For $n \geq 2$,

$D_{dr} \models PGoal(D_1, \text{Starts}(s) \land \text{Init}(s), n, S_0)$.

Since not all $G$-accessible paths are realistic in the sense that they start with a $K$-accessible situation, to filter the unrealistic paths out, KL defined realistic p-goal accessible paths:

$G_R(d, p, n, s) \overset{def}{=} G(d, p, n, s) \land \exists s'. \text{Starts}(p, s') \land K(d, s', s)$.

$G_R$ prunes out the paths from $G$ that are known to be impossible, and since intentions are defined in terms of realistic p-goals, this ensures that these are realistic.

Using realistic p-goals-accessible paths, KL defined intentions as the realistic and maximal consistent prioritized intersection of the agent’s goal hierarchy. First they specify all paths p that are in this prioritized intersection $G_R(d, p, n, s)$:

$G_R(d, p, n, s) \equiv$

if $(n = 0)$

if $\exists p'. G_R(d, p', n, s)$ then $G_R(d, p, n, s)$

else $\exists s'. \text{Starts}(p, s') \land K(d, s', s)$

else

if $\exists p'. (G_R(d, p', n, s) \land G_R(d, p', n - 1, s))$

then $(G_R(d, p, n, s) \land G_R(d, p, n - 1, s))$

else $G_R(d, p, n - 1, s)$.

Using this, they defined what it means for an agent to have an intention at some level n:

$Int(d, \phi, n, s) \overset{def}{=} \forall p. G_R(d, p, n, s) \supset \phi[p]$,

i.e. an agent d has the intention at level n that $\phi$ in situation s if $\phi$ holds over all paths that are in the prioritized intersection of d’s set of $G_R$-accessible paths up to level n in s. Finally, intentions are defined in terms of intentions at n:

$Int(d, \phi, n, s) \overset{def}{=} \forall n. Int(d, \phi, n, s)$,

i.e. the agent d has the intention that $\phi$ in s if for any level n, $\phi$ is d’s intention at n in s.

In our example, given the axioms above, since initially $\phi_0$ and $\phi_1$ are both realistic and are consistent with each other, we can show that initially $D_1$ has the intention that $\phi_0$ and that $\phi_1$:

Proposition 2.

$D_{dr} \models \text{Int}(D_1, \phi_0 \land \phi_1, S_0)$.

Goal Dynamics An agent’s goals change when its knowledge changes as a result of the occurrence of an action, including exogenous events, or when it adopts or drops a goal. KL showed how this can be formalized by specifying how p-goals change. Intentions are then computed using realistic p-goals in every new situation as above.

Since for our example we only need to model cooperative agents that always respect the controller agent’s requests, to simplify, we will modify KL’s framework slightly by introducing a request communicative action and by getting rid of the actions for goal adoption and dropping. req(d, d’, $\phi$) can be used by an agent d to request to adopt a p-goal $\phi$ to another agent $d’$. The APA for this is as follows:

$\text{Poss}(\text{req}(d, d’, \phi), s) \equiv$

$-\text{Int}(d, \lnot \exists s’. p'. \text{Starts}(s') \land$ $\text{Suffix}(p', \text{do}(\text{req}(d, d’, \phi), s)) \land \phi[p'], s)$.

That is, an agent d can request another agent $d’$ to adopt the p-goal $\phi$ if d does not intend in s that it is not the case that it executes the req action next and $\phi$ holds afterwards.

In the following, we specify the dynamics of p-goals by giving the modified SSA for G and discuss each case, one at a time:

$G(d, p, n, \text{do}(a, s)) \equiv$

$\forall d’. \phi.(a \neq \text{req}(d, d’, \phi) \land \text{Progressed}(d, p, n, a, s))$

$\lor \exists d’. \phi. (a = \text{req}(d, d’, \phi) \land \text{Requested}(d, p, n, a, s))$.

6 The construct if $\phi$ then $\delta_1$ else $\delta_2$ is an abbreviation for $(\phi \supset \delta_1) \land (\lnot \phi \supset \delta_2)$.

7 KL used the term “chosen goals” (C-Goals) for this.
For simplicity, we assume that the requested goal is always adopted as the highest priority goal. Other sophisticated models, e.g. one where the requestee adopts the requested goal only if it is from a trusted source, is consistent with its own set of core goals, and at just below these core goals, could have been modeled as easily.

Any path over which the next action performed is not a is eliminated from the respective G-accessibility level for 𝑑.

Secondly, to handle the request of a p-goal 𝜙 directed to 𝑑, we add a new p-goal level containing the requested p-goal 𝜙 to 𝑑’s goal hierarchy at the highest priority by modifying the G-relation accordingly. The G-accessible paths for 𝑑 at level 0 are the ones that share the same history with 𝑑(𝑎, 𝑠) and over which 𝜙 holds. The G-accessible paths for 𝑑 at all levels below 0 are the ones that can be obtained by progressing the level immediately above it. Thus the agent 𝑑 acquires the p-goal that 𝜙 at the highest priority level 0, and all the p-goals in 𝑠 are pushed down one level in the hierarchy.

Proposition 3.

\[ \text{Progressed}(d, p, n, a, s) \triangleq \exists s'. \text{Starts}(p, s') \wedge \text{SameHist}(s', \text{do}(a, s)) \wedge \phi[p] \]

In our example, we can show that the agent 𝐷₁ will have the intention that \( \diamond \text{Vis}(D_1, L_1') \) after 𝐷₁ takes off from 𝐿₅, 𝐷ᵦ informs 𝐷₁ that there is a thunderstorm at 𝐿₁, and 𝐷ᵦ requests 𝐷₁ to eventually visit 𝐿₁', starting in 𝑆₀, i.e. in situation 𝑆₃ = \text{do}(\text{takeOff}(D_1, L_1)); \text{inform}(D_0, D_1, TStorm (L_1)); \text{req}(D_0, D_1, \diamond \text{Vis}(D_1, L_1'))], S₀); thus:

Proposition 4.

\[ D_{\text{er}} \models \text{Int}(D_1, \diamond \text{Vis}(D_1, L_1'), S₃). \]

But 𝐷₁ will not have the intention that 𝜙₁ as it has become impossible for 𝐷₁ to visit 𝐿₁ due to its knowledge of the thunderstorm at 𝐿₁, i.e.:

Proposition 4.

\[ D_{\text{er}} \models \sim \text{Int}(D_1, \phi_1, S₃). \]

Proving the above two propositions involve progressing 𝐷₁’s G-accessible paths using the SSA for \( G \) and then recomputing its intentions in \( S₃ \) using the definition of \( \text{Int} \) and the axiom for \( G_{\text{r}} \).

4 Handling Conative Effects

Given a trace of events, actual achievement causes are the events that are behind achieving an effect. ⁹ To formalize reasoning about epistemic effects, KL [22] introduced the notion of epistemic dynamic formulae in the SC. An effect in their framework is thus an epistemic dynamic formula. We will extend this notion to that of intentional dynamic formulae \( \varphi \) to deal with conative effects (see below). Given an effect \( \varphi \), the actual causes are defined relative to a narrative (variously known as a scenario or a trace) \( s \). When \( s \) is ground, the tuple \( \langle \varphi, s \rangle \) is often called a causal setting [3]. Also, it is assumed that \( s \) is executable, and \( \varphi \) is false before the execution of the actions in \( s \), but became true afterwards, i.e. \( D \models \text{executable}(s) \wedge \sim \varphi(\text{root}(s)) \wedge \varphi(s) \), where \( \text{root}(s) = \text{root}(s') \), if \( \exists a'. s = \text{do}(a', s') \), and \( \text{root}(s) = s \), otherwise. Here \( \varphi(s) \) denotes the formula obtained from \( \varphi \) by restoring the appropriate situation argument into all fluents in \( \varphi \) (see Definition 2).

Note that since all changes in the SC result from actions, the potential causes of an effect \( \varphi \) are identified with a set of action terms occurring in \( s \). However, since \( s \) might include multiple occurrences of the same action, one also needs to identify the situations where these actions were executed. To deal with this, KL required that each situation is associated with a time-stamp. Since in the context of knowledge, we will have different \( K \)-accessible situations where an action occurs, using time-stamps provides a common reference/tigid designator for the action occurrence. The initial situations start at time 0 and each action increments the time-stamp by one. Thus, our theory includes the following axioms:

\[ \text{Init}(s) \supset \text{time}(s) = 0, \]
\[ \forall a, s.t. \text{time}(\text{do}(a, s)) = t \equiv \text{time}(s) = t - 1. \]

With this, causes in this framework is a non-empty set of action-time-stamp pairs derived from the trace \( s \) given \( \varphi \).

We now introduce our notion of intentional dynamic formulae (IF, henceforth):

Definition 1. Let \( \bar{x}, \theta_a, \text{and } \bar{y} \) respectively range over object terms, action terms, and object and action variables. The class of situation-suppressed intentional dynamic formulae \( \varphi \) is defined inductively using the following grammar:

\[ \varphi ::= P(\bar{x}) | \text{Poss}(\theta_a) | \text{After}(\theta_a, \varphi) | \sim \varphi | \varphi_1 \wedge \varphi_2 | \exists y. \varphi | \text{Know}(\text{agt}, \varphi) | \text{Int}(\text{agt}, \psi). \]

That is, an IF can be a situation-suppressed fluent, a formula that says that some action \( \theta_a \) is possible, a formula that some IF holds, a formula that can be built from other IF using the usual connectives, or a formula that the agent knows a formula that some IF holds or intends to bring about some path formula \( \psi \). Note that \( \varphi \) can have quantification over situations and action variables, but must not include quantification over situations or ordering over situations (i.e. \( \Box \) or arbitrary \( K \) or \( G \)-relations, i.e. those that do not come from the expansion of \( \text{Know}/\text{Int} \). We will use \( \varphi \) for IF.

Note that the argument of \( \text{Int} \) in the above inductive definition is a path formula \( \psi \). Thus to allow for IF in the context of \( \text{Int} \), we need to redefine state formula \( \Phi \) to include IF \( \varphi \):

\[ \Phi ::= P(\bar{x}) | \text{A}\phi | \Phi \wedge \Phi | \sim \Phi | \forall x. \Phi | \varphi. \]

Also, the following addition to the definition of \( \Phi[\cdot] \) is needed:

\[ \Phi[s] \triangleq \varphi(s), \text{ if } \Phi \text{ is of the form } \varphi. \]

We define \( \varphi(\cdot) \) as follows:
We will now present the definition of causes in the SC. The idea behind how causes are computable is as follows. Given an effect \( \varphi \) and scenario \( s \), if some action of the action sequence in \( s \) triggers the formula \( \varphi \) to change its truth value from false to true relative to \( D \), and if there are no actions in \( s \) after it that change the value of \( \varphi \) back to false, then this action is an actual cause of achieving \( \varphi \) in \( s \). Such causes are referred to as primary causes.

**Definition 3 (Primary Cause).**

\[
\text{CausesDirectly}(a, t, \varphi, s) \iff \\
\exists s_a. \text{time}(s_a) = t \land (\text{root}(s) < \text{do}(a, s_a) \leq s) \\
\land \neg \varphi(s_a) \land \forall s'. (\text{do}(a, s_a) < s' \leq s \supset \varphi(s')).
\]

That is, \( a \) executed at time \( t \) is the primary cause of effect \( \varphi \) in situation \( s \) iff \( a \) was executed in a situation with time-stamp \( t \) in scenario \( s \), a caused \( \varphi \) to change its truth value to true, and no subsequent actions on the way to falsify \( \varphi \).

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 on the trace that contributed to the preconditions and the context conditions of a cause must be considered as a cause as well. The following definition captures both primary and indirect causes.

**Definition 4 (Actual Cause [22]).**

\[
\text{Causes}(a, t, \varphi, s) \equiv \\
\forall P \forall t, s, t, \varphi. \left( \text{CausesDirectly}(a, t, \varphi, s) \supset P(a, t, \varphi, s) \right) \\
\land \forall a, t, s, \varphi. \left( \text{CausesDirectly}(a, t', \varphi, s) \land \text{time}(s') = t' \land s' < s \right) \\
\land \land P(a, t, \left[ \text{Poss}(a') \land \text{After}(a', \varphi) \right], s') \\
\supset P(a, t, \varphi, s)
\]

Thus, \( \text{Causes} \) is defined to be the least relation \( P \) such that if \( a \) executed at time \( t \) directly causes \( \varphi \) in scenario \( s \) then \( (a, t, \varphi, s) \) is in \( P \), and if \( a' \) executed at \( t' \) is a direct cause of \( \varphi \) in \( s' \), the time-stamp of \( s' \) is \( t' \), \( s' < s \), and \( (a, t, \left[ \text{Poss}(a') \land \text{After}(a', \varphi) \right], s') \) is in \( P \) (i.e. \( a \) executed at \( t \) is a direct or indirect cause of \( \left[ \text{Poss}(a') \land \text{After}(a', \varphi) \right] \) in \( s' \)), then \((a, t, \varphi, s)\) is in \( P \). Here the effect

\( \text{Poss}(a') \land \text{After}(a', \varphi) \) requires \( a' \) to be executable and \( \varphi \) to hold after \( a' \).

With these simple modifications, the framework is now capable of dealing with conative effects. To see this, consider the following scenario \( \sigma \) in our example, where \( \sigma = \text{do}(\text{takeOff}(D_1, L_s)); \text{inform}(D_1, D_2, \text{TStorm}(L_1)); \text{req}(D_1, D_2, \text{Vis}(D_1, L_1')) + \text{flyTo}(D_1, L_1, L_1'); \text{flyTo}(D_1, L_2, L_4)), S_0 \). There are 7 actions in this scenario. For convenience, we will use \( \alpha_i \) to denote the first \( i \) actions in this trace, and so do([\alpha_i]), \( S_0 \) is the situation obtained from executing the first 5 actions starting in \( S_0 \). Now assume that we want to reason about the causes of the effect \( \varphi_1 = \text{Int}(D_1, \text{Vis}(D_1, L_1')) \) in scenario \( S_1 = \text{do}(\alpha_5), S_0 \). Then we can show that:

**Proposition 5.**

\[ D_{de} \models \text{Causes}(\text{req}(D_1, D_1, \text{Vis}(D_1, L_1')), 2, \varphi_1, \sigma_1), \]

i.e. as expected, \( D_1 \)'s request to \( D_1 \) to eventually visit \( L_1' \) that was executed at time 2 is the cause of \( D_1 \)'s intention that \( \text{Vis}(D_1, L_1') \).

## 5 Reasoning about Agent Behaviour

We are now ready to formalize reasoning about agent behaviour via causation. Just like causes, an explanation in our framework is also modeled using an action-time-stamp pair \((a, t)\). Agent behaviour, on the other hand, is captured using a situation \( s \) and relative to an observation \( \varphi \). For this, we use the predicate \( \text{Explains}(a, t, \varphi, s) \), which means that the action \( a \) executed at time \( t \) explains the behaviour of the agents captured in situation \( s \) relative to the observation \( \varphi \). For example, \( \text{Explains}(a_{dr}, t_{dr}, \varphi_2, \sigma) \) states that the behaviour of drones as modeled by situation/scenario \( \sigma \) relative to the effect that \( \varphi_2 = \text{Vis}(D_1, L_1') \) can be explained by action \( a_{dr} \) executed at time \( t_{dr} \) (see below for the values/binding of \( a_{dr} \) and \( t_{dr} \)). Thus, \( (a_{dr}, t_{dr}) \) explains why the drone \( D_1 \) visited the location \( L_1' \). Note that, just as in the case for achievement causation, we assume here that \( \neg \varphi(\text{root}(s)) \land \varphi(s) \).

While explaining agent behaviour through the causes of the effect is reasonable, it may not always be insightful. For instance, we can argue that the behaviour of the drone \( D_1 \) in \( s \) w.r.t. visiting \( L_1 \) can be explained by its action \( \text{flyTo}(D_1, L_2, L_1') \) executed at time 5. However, this is obvious and is not very useful. A deeper level of explanation requires analyzing the motivations of the involved agents, in particular their intentions behind executing the actions that caused the effect.

To further explain agent behaviour, we will use an intention recognition component, which for this paper is considered to be a black-box module. We use the predicate \( \text{RRInt}(d, \phi, a, t, s) \) to denote that agent \( d \) is recognized to have the relevant intention that \( \phi \) in situation \( s \) w.r.t. the action \( a \) executed at time \( t \). For instance, \( \text{RRInt}(D_1, \text{Vis}(D_1, L_1'), \text{flyTo}(D_1, L_2, L_1'), 5, \sigma) \) says that in scenario \( \sigma \), agent \( D_1 \) is recognized to have the intention that \( \text{Vis}(D_1, L_1') \) for executing the action \( \text{flyTo}(D_1, L_2, L_1') \) at time 5. With this, we can further explain an agent’s behaviour via the root-cause analysis of its intentions behind performing actions. In our example, since \( D_1 \) flew to \( L_1' \) due to its intention that \( \text{Vis}(D_1, L_1') \), it is reasonable to explain its behaviour via the causes of having this intention in the first place. This will reveal that \( D_1 \) had this intention due to \( D_1 \)'s request, and thus its behaviour w.r.t. visiting \( L_1' \) can be explained by this request action.

We now give the definition for \( \text{Explains} \):
Definition 5.

\[ \text{Explains}(a, t, \varphi, s) \]  
\[ \equiv \forall P. \forall a, t, s, \varphi. (\neg \varphi(\text{root}(s)) \land \varphi(s) \land \text{Causes}(a, t, \varphi, s) \land P(a, t, \varphi, s)) \land \exists a', t', d, s', \psi. (P(a', t', \varphi, s) \land \text{agent}(a') = d' \land \text{RRInt}(d, \psi, a', t', s) \land \text{time}(s') = t' \land s' < s \land \neg \text{Int}(d', \psi, \text{root}(s')) \land \text{Int}(d', \psi, s') \land \text{Causes}(a, t, \text{Int}(d', \psi), s')) \land P(a, t, \varphi, s). \]

Thus, \( \text{Explains} \) is defined to be the least relation \( P \) such that if an action \( a \) executed at time \( t \) causes \( \varphi \) in scenario \( s \), then \( (a, t, \varphi, s) \) is in \( P \), and if \( (a', t', \varphi', s') \) is in \( P \) (i.e. some other action \( a' \) executed at time \( t' \) explains \( \varphi \) in \( s' \)), the agent of \( a' \) is \( d' \), \( d' \) is recognized to have the intention that \( \psi \) behind performing \( a' \) at \( t' \) in \( s' \), the timestamp of \( s' \) is \( t' \), \( s' < s \), and \( a \) executed at \( t \) was the cause of this intention in \( s' \), then \( (a, t, \varphi, s) \) is in \( P \). Here \text{agent}(a) denotes the agent of the action \( a \); it can be specified as usual by an axiom that returns the agent of \( a \), usually the first argument of \( a \), i.e. \text{agent}(a(d, \vec{x})) = d. \) Also, \( s' \) is the situation where \( a' \) was executed. Finally, the two requirements that the effect be false before the execution of the actions in the scenario and becomes true afterwards, i.e. \( \neg \varphi(\text{root}(s)) \land \varphi(s) \land \text{Causes}(a, t, \varphi, s) \), and \( \neg \text{Int}(d', \psi, \text{root}(s')) \land \text{Int}(d', \psi, s') \), are needed to ensure that the achievement causes actually exist.

Put otherwise, agent behaviour relative to the observation that \( \varphi \) in scenario \( s \) can be explained by the action \( a \) executed at time \( t \) iff \( a \) at \( t \) is a cause of \( \varphi \) in \( s \), or at \( t \) causes the intention behind some explanation of \( \varphi \) in \( s \).

Returning to our example, we can now formally state the two explanations that we mentioned above and give the bindings for \( a_{dr} \) and \( t_{dr} \). First, we can show that:

**Proposition 6.**

\( D_{dr} \models \text{Explains}((flyTo(D_{1}, L_s, L_1)) \cup \text{flyTo}(D_{1}, L_s, L_1', 3), 5, \varphi_2, \sigma). \)

But perhaps more interestingly, we can show that:

**Proposition 7.**

\( D_{dr} \models \text{Explains}(\text{req}(D_c, D_1, \text{flyTo}(D_{1}, L_s, L_1)), 2, \varphi_2, \sigma), \)

where, \( D_{dr} \models \text{Explains}(\text{req}(D_c, D_1, \text{flyTo}(D_{1}, L_s, L_1)), 2, \varphi_2, \sigma). \)

It is important to note that the scenario \( s \) in Definition 5 may and will often include the actions of multiple agents, and thus explanation of agent behaviour may trigger the analysis of the mental states of multiple agents. For example, given another suitable scenario, recognizing the intention behind the controller agent \( D_c \)’s request to \( D_1 \) and analyzing this intention might have in turn exposed the causes behind its actions, e.g. due to its prior commitments to safety, etc. As such, the analysis performed here is truly multiagent in nature. Also, although our example only involves single-action/direct causes and we do not consider epistemic effects, as discussed above, the framework does support secondary causes and causal knowledge dynamics; see [22] for concrete examples of these.

A potential issue with the above definition of explanation is that in some domains, it might produce counter-intuitive results. Indeed, it is not very hard to come up with examples where the causes of the intention behind executing secondary and tertiary (i.e. non-direct) causes might be irrelevant. For example, a precondition of the \( \text{flyTo}(D_1, L_s, L'_1) \) action might have been that the drone \( D_1 \) has enough fuel, and thus refueling \( D_1 \) might have been an indirect cause of eventually visiting \( L'_1 \); however the causes of the intention behind refueling \( D_1 \) have nothing to do with visiting \( L'_1 \). Thus, while analyzing the causes of the intention behind a primary or direct cause seems useful, this is not always the case for indirect causes. To deal with this, the following variant of \( \text{Explains} \) can be adopted:

\[ \text{Explains}(a, t, \varphi, s) \]  
\[ \equiv (\neg \varphi(\text{root}(s)) \land \varphi(s) \land \text{Causes}(a, t, \varphi, s) \lor \text{ExplainsDirectly}(a, t, \varphi, s)), \]

where \( \text{ExplainsDirectly} \) is just like \( \text{Explains} \) but with \( \text{Causes} \) replaced with \( \text{CausesDirectly} \).

6 Properties

We next discuss some general properties of our formalization. We start by showing that requests work as expected:

**Theorem 1 (Adoption).**

\( D \models \neg \text{Know}(\text{agt}, \neg \exists p, p'. \text{Starts}(p, \text{now}) \land \text{Suffix}(p', \text{p, do(\text{req(agt, \text{agt, \phi}, \text{now})})} \land \phi[p', \text{s}]) \lor \text{Int}(\text{agt, \phi, \text{do(\text{req(agt', \text{agt, \phi, \phi'})})}). \)

**Proof Sketch.** Since by the antecedent and the SSA for \( K \), there is a path that starts with a \( K \)-accessible situation in \( \text{do(\text{req(agt', \text{agt, \phi, \phi'})})} \) which \( \phi \) holds, by the SSA for \( G \) and the axiom for \( G_{\cap} \), the \( G_{\cap} \)-accessible paths at level 0 in situation \( \text{do(\text{req(agt', \text{agt, \phi, \phi'})})} \) will only include paths over which \( \phi \) holds. The rest of the proof follows from this, the definition of \( \text{Int} \), and the axiom for \( G_{\cap} \). \( \square \)

That is, an agent \( \text{agt} \) acquires the intention that \( \phi \) when requested by another agent \( \text{agt}' \) that \( \phi \) in situation \( s \) provided that \( \phi \) is not known to be impossible after the request action has happened in \( s \). Note that this holds even if \( \phi \) is inconsistent with \( \text{agt}' \)'s current intentions as the requested goal is always adopted at the highest priority level (again, this is by design). However, at all times the agent’s intentions remain consistent. Thus if some other intention \( \phi'^{-} \) at some priority level in \( s \) is inconsistent with the newly adopted goal \( \phi, \phi'^{-} \) is simply ignored when computing the intentions in \( \text{do(\text{req(agt', \text{agt, \phi, \phi'})})} \).

Next, we turn our attention to properties of explanations. Let \( D' = D \cup D_{RRInt} \), where \( D_{RRInt} \) is a fixed set of \( RRInt(\text{sentences}) \). First, note that explanations include causes, i.e.:

**Corollary 1.**

\( D' \models \text{Causes}(a, t, \varphi, s) \lor \text{Explains}(a, t, \varphi, s). \)

Moreover, we can show that explanations are not necessarily causes. This can be proven using Proposition 7 and the following lemma, which together produce a counterexample.

**Lemma 1.**

\( D_{dr} \models \neg \text{Causes}(\text{req}(D_c, D_1, \text{flyTo}(D_{1}, L_s, L_1)), 2, \varphi_2, \sigma). \)

We next show that explaining agent behaviour relative to logically equivalent effects has the same result:
Proposition 8 (Extensionality w.r.t. Effects),
\[ D^* \models (\forall a, t, s. \text{Explains}(a, t, \varphi, s) \equiv \text{Explains}(a, t, \varphi_2, s)). \]

Proof. This follows from the fact that we are using a possible worlds/paths semantics.

Finally, we study the conditions under which action occurrences do not alter knowledge about explanations.

Theorem 2 (Persistence of Knowledge about Explanations),
\[ D^* \models (\forall a, t, s. \text{Explains}(a, t, \varphi, s) \equiv \text{Explains}(a, t, \varphi_2, s)) \]

Proof Sketch. We first show that the agent’s causal knowledge remains unchanged when the antecedent holds. This can be shown by proving that the causes of \( \varphi \) remain the same in every epistemic alternative as \( \varphi \) remains true in every new situation. Since \( D_{\text{new}} \subseteq D \) does not change, the rest of the proof then follows from this and the definition of \text{Explains}.

That is, if an agent knows in \( s \) whether an action \( a \) executed at time \( t \) is a cause of an effect \( \varphi \) (and thus whether \( a \) in \( t \) explains \( \varphi \) in \( s \)), it will continue to know whether \( a \) executed at \( t \) explains \( \varphi \) in a future situation \( s' \), provided that its knowledge of the effect \( \varphi \) does not change between \( s \) and \( s' \).

However, this is not the case in general. For instance, if the agent ceases to know that \( \varphi \), then in this new situation it will not know what actions are causes, and consequently neither what actions explain \( \varphi \).

7 Related Work and Conclusion

In this paper, we proposed a formal account of causal reasoning about motivations. Using this, we offered a novel take on explainable AI that is grounded in theory of mind: agent behaviour in our framework can be explained via the causal analysis of observed effects, which in turn can trigger the analysis of their mental states.

Recently, the pursuit of transparent and explainable AI systems has led to a renewed interest in the study of cognitive aspects of knowledge representation (KR), as advocates of KR often argue that its declarative nature makes it cognitively more suitable for explanation purpose. There has been some work on formalizing explanation in KR. For instance, in his early work, Shanahan [36] proposed a deductive and an abductive approach to explanation in the situation calculus, both of which are based on default reasoning. More recently, Shvo et al. [39] proposed a belief revision-based account of explanation. In their framework, a formula \( \phi \) explains another formula \( \psi \) if revising by \( \phi \) makes the agent believe \( \psi \) and the agent’s beliefs remain consistent afterwards. In [6], Dennis and Oren used dialogic between the user and a Belief-Desire-Intention (BDI) agent system to explain why the agent has chosen a particular action. Their approach aims to identify any divergence of views that exist between the user and the BDI agent relative to the latter’s behaviour and allows for an interactive and user-friendly explanation process. In his SEM-based formalization, Beckers [4] presented formal definitions of various causal notions of explanation and proposed to use actual causation for the purpose of explainable AI. The connections between these notions and the consequences of ignoring the causal structure are explored. Miller [29] proposed a contrastive explanation model based on structural causal models to enhance understanding and trust in AI decision-making. In [27], SEM-based causal models are utilized to generate explanations of the behaviour exhibited by model-free reinforcement learning agents. Finally, Sridharan et al. [41, 42] proposed an explainable robotic architecture by integrating step-wise refinement, non-monotonic reasoning, probabilistic planning, and interactive learning. However, none of the aforementioned work formalize causal analysis of agent motivation or employ such reasoning along with theory of mind for explaining agent behaviour (while Shvo, Klassen, and McIlraith [39] appealed to theory of mind, they did not address actual causation). In fact to the best of our knowledge, our proposal is the first and the only attempt to this end.

Our current formalization is limited in many ways. For instance, our proposal does not comply with many of the desiderata for explanations proposed by [39], in particular those that are related to belief and belief revision, since we only support knowledge and knowledge update. Also, we only allow deterministic and fully observable actions. Scenarios in our framework are linear, i.e. we assume that the order of action occurrence is known. This also means that while our proposal supports multiple agents, the underlying framework assumed by our work must ensure that these agents only act one at a time. When dealing with causation and explanations, we computed achievement causes only. In the literature, other types of causes has also been studied, e.g., actual maintenance causes; these are responsible for mitigating the threats to the achieved effect [2]. Incorporating other types of causes thus would have allowed us to explain effects further and in finer details. We leave these for future work.

Acknowledgements

We would like to thank the anonymous reviewers of ECAI, CARK, and XLoKR 2023 for directing us to the relevant literature and helping us improve this paper considerably. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number RGPIN-2022-03433].

References
