Contextual Representations of Cause via Reasoning about Actions and Change

EMILY C. LEBLANC, MARCELLO BALDUCCINI
College of Computing and Informatics, Drexel University, Philadelphia, PA

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

In Causality, *actual cause* can be defined as an event determined to be responsible for bringing about a given outcome in a specific scenario. For example, Socrates drinking hemlock can be viewed as the actual cause of the philosopher's death. However, it is also reasonable to consider why he consumed it in the first place — if he was forced to drink the hemlock as punishment for a crime, then is his acting against the law another cause of his death? Clearly, an outcome can be associated with multiple actual causes and choosing the most suitable among them often requires considering the context of the inquiry. In this paper, we introduce the concept of *cause with context* (CwC), a variation on the notion of actual cause that considers contextual information about a cause's relationship to the domain in which it has occurred. We present a novel framework for representing CwCs, inspired by Counterfactual Reasoning and making use of techniques from Reasoning about Actions and Change to support reasoning over domains that change over time. We also present an approach for computing CwCs via Answer Set Programming.

KEYWORDS: causality, counterfactual reasoning, reasoning about actions and change, knowledge representation and reasoning, answer set programming

Introduction

In Causality, actual cause can be defined as an event determined to be responsible for bringing about a given outcome in a specific scenario (Pearl 2009). When humans inquire about an outcome, determining the actual cause often requires considering the context of the query. Consider a simplified story outlining the events of the trial of Socrates: Socrates questioned the gods recognized by his city, was charged with impiety, was sentenced to death, and drank poisonous hemlock in accordance with his sentence. If we were to ask experts in the fields of medicine, law, and philosophy to determine the cause of Socrates' death with respect to the context of their expertise, we might receive the following answers:

- Medical context: *Socrates drank poisonous hemlock.*
- Legal context: *Socrates was sentenced to death.*
- Philosophical context: *Socrates questioned the deities recognized by his city.*

Each answer corresponds to a different event in the story and each is reasonable with respect to its context. This thought exercise reconfirms the well-known fact that an outcome of interest (here, Socrates' death) may be associated with multiple actual causes (Pearl 2009). Moreover, it confirms our position that selecting the most suitable cause for an outcome of interest can require consideration of the query's context. This is the primary motivation of our work.

In this paper, we introduce the concept of a *cause with context* or CwC (pronounced "quick"), a variation on the notion of actual cause that considers contextual information about a cause's relationship to the outcome of interest and the domain in which it has occurred. We present a novel framework for defining CwCs inspired by the counterfactual definition of cause (Lewis 1973), discussed in greater detail in the Background section of this paper. Our approach makes use of techniques from Reasoning about Actions and Change (RAC) (McCarthy and Hayes 1969) to support reasoning over domains that change over time.

The organization of the paper is as follows. In the next section, we discuss the counterfactual definition of cause and RAC as they pertain to our approach. Following that, we provide background for our formalization of knowledge and events. Next, we present the technical details of our framework and demonstrate its ability to represent CwCs using the example of Socrates' death. We then present an approach to computing CwCs via Answer Set Programming. Following that, we present a summary of related work. Finally, we draw conclusions and discuss directions for future research.

Background

The Counterfactual Definition of Cause. Counterfactual reasoning is the process of reasoning about the consequences of events, circumstances, or scenarios that have not occurred. In this work, we are concerned with reasoning counterfactually about scenarios that are contrary to observations that we have gathered. The counterfactual definition of cause (Lewis 1973) is based on the idea that if *x* is a *necessary cause* of *y* and *x* does not occur, then *y* will not occur. From a reasoning perspective, this suggests that if we can determine that the counterfactual *"If x had not occurred, then y would not have occurred"* is true, then it is possible to claim that *x* has caused *y* in a particular scenario. The relationship of counterfactuals and cause, particularly actual cause, have been the subject of much study in philosophy (Lewis 1973; Collins and Hall 2004) and in Artificial Intelligence (Ginsberg 1986; Pearl 2009; Pereira et al. 1991; Halpern and Hitchcock 2011; Vennekens et al. 2010; Merck and Kleinberg 2016).

Reasoning about Actions and Change (RAC). RAC is concerned with representing the properties of actions (McCarthy and Hayes 1969). Research in this field studies reasoning over domain knowledge and, specifically, about the direct and indirect effects of actions, and has uncovered a variety of interesting representation and reasoning problems (Lifschitz 1987; Haugh 1987; Gelfond and Lifschitz 1993; Lin 1996; McCain et al. 1997; Thielscher 1997). In our work, we aim to leverage mentions of observed events and domain knowledge to determine a detailed picture of a scenario over time. The section on Preliminaries provides a more technical discussion of RAC with respect to the description of domains.

Preliminaries

Action Language $\mathcal{AL}$. For the representation of the domain and of its evolution over time we rely on action language $\mathcal{AL}$ (Baral and Gelfond 2000). The syntax of $\mathcal{AL}$ builds upon an alphabet consisting of a set $\mathcal{F}$ of symbols for *fluents* and a set $\mathcal{E}$ of symbols for *events*[1].

Fluents are boolean properties of the domain, whose truth value may change over time. A

[1] For convenience and compatibility with the terminology from RAC, in this paper we use *action* and *event* as synonyms.

fluent literal is a fluent f or its negation $\neg f$. Additionally, $\overline{f} = \neg f$ and $\overline{\neg f} = f$ A statement of the form

$$e \text{ causes } l_0 \text{ if } l_1, l_2, \ldots, l_n \tag{1}$$

is called *dynamic (causal) law*, and intuitively states that, if event e occurs in a state in which literals $l_1, \ldots, l_n$ hold, then l_0, the *consequence of the law*, will hold in the next state. A statement

$$l_0 \text{ if } l_1, \ldots, l_n \tag{2}$$

is called *state constraint* and says that, in any state in which $l_1, \ldots, l_n$ hold, l_0 also holds. This second kind of statement allows for an elegant and concise representation of side-effects, which increases the expressivity of the language. Finally, an *executability condition* is a statement of the form:

$$e \text{ impossible\_if } l_1, \ldots, l_n \tag{3}$$

where e and $l_1, \ldots, l_n$ are as above. (3) states that e cannot occur if $l_1, \ldots, l_n$ hold. A set of statements of $\mathscr{AL}$ is called *action description*. The semantics of $\mathscr{AL}$ maps action descriptions to transition diagrams, as discussed next.

A set S of literals is *closed under a state constraint* (2) if $\{l_1, \ldots, l_n\} \not\subseteq S$ or $l_0 \in S$. S is *consistent* if, for every $f \in \mathscr{F}$, at most one of f, $\neg f$ is in S. It is *complete* if at least one of f, $\neg f$ is in S. A *state* of an action description *AD* is a complete and consistent set of literals closed under the state constraints of *AD*.

Given an event e and a state σ, the set of *(direct) effects of e in* σ, denoted by $E(e, \sigma)$, is the set that contains a literal l_0 for every dynamic law (1) such that $\{l_1, \ldots, l_n\} \subseteq \sigma$. Given a set S of literals and a set Z of state constraints, the *set* $Cn_Z(S)$ *of consequences of* S *under* Z is the smallest set of literals that contains S and is closed under Z. Finally, an event e is *non-executable* in a state σ if there exists an executability condition (3) such that $\{l_1, \ldots, l_n\} \subseteq \sigma$. Otherwise, the event is *executable* in σ.

The semantics of an action description *AD* is defined by its *transition diagram* $\tau(AD)$, a directed graph $\langle N, E \rangle$ such that:

- N is the collection of all states of *AD*;
- E is the set of all triples $\langle \sigma, e, \sigma' \rangle$ where σ, σ' are states, e is an event executable in σ, and σ, e, σ' satisfy the *successor state equation*:

  $$\sigma' = Cn_Z(E(e, \sigma) \cup (\sigma \cap \sigma'))$$

  where Z is the set of all state constraints of *AD*.

A triple $\langle \sigma, e, \sigma' \rangle \in E$ is called a transition of $\tau(AD)$ and σ' is a *successor state of* σ (under e). As sequence $\langle \sigma_1, \alpha_1, \sigma_2, \ldots, \alpha_k, \sigma_{k+1} \rangle$ is a *path of* $\tau(D)$ of length k if every $\langle \sigma_i, \alpha_i, \sigma_{i+1} \rangle$ is a transition in $\tau(D)$. We refer to state σ_1 of a path p as the *initial state* of p. A path of length 0 contains only an initial state. In the next section, we build upon this formalization to define a framework for defining CwCs.

Representing Causes with Context

In the example of the trial of Socrates, we claimed that a legal expert would tell us that the actual cause of his death is that he was charged with impiety. This conclusion can be reached intuitively by considering that Socrates was sentenced to his death as a consequence of the charge and

that he carried out his sentence by drinking the poison. In this section, we formally define the concepts of a novel framework for representing causes with context and show that the framework is capable of representing a CwC relating Socrates' death to his sentencing.

Assumptions. The goal of our investigation is to develop a framework with which we may represent and reason about CwCs. We assume that we have complete knowledge of direct and indirect effects of any events occurring in the domain. The knowledge is specified by a combination of dynamic laws, state constraints, and executability conditions. We also assume that all knowledge is correct. This assumption simplifies the problem and allows us to disregard issues related to uncertainty and inaccurate information that might lead to questionable conclusions about CwCs.

Representing CwCs. In order to represent CwCs, our framework requires a description of a scenario, an action description for the domain in which the scenario occurred, and the outcome of interest for which a cause must be identified. We assume that these are provided by an external source and therefore we regard this information as constituting a query to the framework. We begin our discussion on the formalization of CwCs by defining the components of a query. A *domain description* consists of an action description reflecting the scenario's domain and of an *event sequence*, i.e. a temporal ordering of the events from the scenario:

Definition 1
A *domain description* is the tuple $DD = \langle v, AD \rangle$ where $v = \langle e_1, e_2, \ldots, e_k \rangle$ is an event sequence containing k events, and AD is an action description.

Recall the events leading up to the death of Socrates in our example: he questioned the gods, was charged with impiety, was sentenced to death, and drank the poison. These events can be encoded as an event sequence as follows:

$$\begin{aligned} v_{soc} = \langle & questionsGods(socrates), \\ & chargedWith(socrates, impiety), \\ & sentencedTo(socrates, death), \\ & drinks(socrates, hemlock) \rangle \end{aligned}$$

The following action description AD_{soc} provides a characterization of events occurring in the Socrates domain:

$$AD_{soc} = \begin{cases} chargedWith(X,Y) \text{ causes } inPrison(X), & (4) \\ escapes(X, prison) \text{ causes } \neg inPrison(X) \text{ if } inPrison(X), & (5) \\ drinks(X, hemlock) \text{ causes } \neg isAlive(X) \text{ if } isAlive(X), & (6) \\ sentencedTo(X,Y) \text{ impossible\_if } \neg inPrison(X), & (7) \\ walksOutside(X) \text{ impossible\_if } inPrison(X) & (8) \end{cases}$$

Some rules of AD_{soc} are straightforward, such as the effect of X drinking hemlock being that X is not alive, and so we will not describe them in detail here. For illustration purposes, AD_{soc} includes a few simplified commonsense rules about the legal domain. Law (7) states that it is impossible for X to be sentenced if X is not in custody. Law (8) states that it is impossible for

Table 1: *Table representation of the factual path f_{soc} in the trial of Socrates example.*

State	Event
$\sigma_1 = \{isAlive(socrates), \neg inPrison(socrates)\}$	*questionsGods(socrates)*
$\sigma_2 = \{isAlive(socrates), \neg inPrison(socrates)\}$	*chargedWith(socrates, impiety)*
$\sigma_3 = \{isAlive(socrates), inPrison(socrates)\}$	*sentencedTo(socrates, death)*
$\sigma_4 = \{isAlive(socrates), inPrison(socrates)\}$	*drinks(socrates, hemlock)*
$\sigma_5 = \{\neg isAlive(socrates), inPrison(socrates)\}$	—

someone to take a walk outside (of the prison) when they are imprisoned. Given v_{soc} and AD_{soc}, the domain description for the Socrates scenario is $DD_{soc} = \langle v_{soc}, AD_{soc} \rangle$.

As we mentioned earlier, we also assume that we are provided with a consistent set of fluent literals θ representing an *outcome* of interest. In our example, we are interested in representing a cause for Socrates's death, so the outcome is $\theta_{soc} = \{\neg isAlive(socrates)\}$ in accordance with the fluents in AD_{soc}.

Definition 2
A *query* is a tuple $\mathscr{Q} = \langle v, AD, \theta \rangle$ where v and AD are the elements of a domain description DD and θ is an outcome.

The query for our example is $\mathscr{Q}_{soc} = \langle v_{soc}, AD_{soc}, \theta_{soc} \rangle$. Given a query, we build upon its components in order to define CwCs. We begin this process by representing the set of all possible mappings of the event sequence v from the domain description to paths of the transition diagram $\tau(AD)$. We refer to these mappings as *factual paths*.

Definition 3
Given a query $\mathscr{Q} = \langle v, AD, \theta \rangle$ such that $v = \langle e_1, e_2, \ldots, e_k \rangle$, a *factual path* is a path $f = \langle \sigma_1, \alpha_1, \sigma_2, ..., \alpha_k, \sigma_{k+1} \rangle$ of $\tau(AD)$ satisfying the following conditions:

1. $\forall i, 1 \leq i \leq k,\ \alpha_i = e_i$
2. $\theta \subseteq \sigma_{k+1}$

Note that, because of potentially non-deterministic effects of actions in AD,[2] an event sequence v may map to multiple paths in $\tau(AD)$. In the following discussion, the set of all factual paths with respect to some domain description $DD = \langle v, AD \rangle$ and outcome θ is denoted by $F = \{f_1, f_2, \ldots, f_m\}$.

Condition 1 requires that the events in f correspond to the events of v in order of their occurrence, capturing the idea that each event of v corresponds to a transition between states in f. Condition 2 requires that the fluent literals of the outcome θ are satisfied in the final state of f, capturing the idea that the outcome is expected to hold in the final state of every factual path. Table 1 provides a representation of a factual path f_{soc} that corresponds to query $\mathscr{Q}_{soc}$ given above.

[2] For example, consider the action description $\{q$ if $\neg r, p;\ r$ if $\neg q, p;\ a$ causes $p\}$.

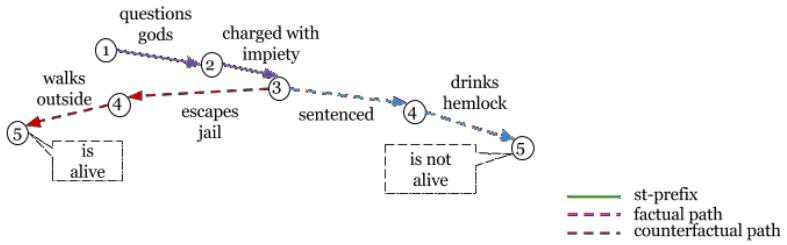

Figure 1: Factual and counterfactual paths for reasoning about Socrates' death.

The first column of the table lists the values of fluents in each state σ_i of f_{soc} and the second column gives the event α_i that causes a transition to the state σ_{i+1} given in the following row. It is easy to see that the factual path f satisfies both conditions with respect to v_{soc} and θ_{soc}.

In our approach, the process of reasoning about actual cause requires finding *counterfactual paths* in $\tau(AD)$ that diverge from a given factual path f with the occurrence of some "counterfactual event" and terminates in a state where the outcome of interest does not hold. A (possibly empty) set of counterfactual paths exists for every factual path $f \in F$ with respect to a query $\mathscr{Q}$. Representing a counterfactual path and its divergence from f requires, in part, identifying states and events shared by both paths prior to the point at which they diverge. We formalize this notion as follows.

Given a path $p = \langle \sigma_1, \alpha_1, \sigma_2, \ldots, \alpha_k, \sigma_{k+1}, \rangle$, we call the sequence $\rho = \langle \sigma_1, \alpha_1, \sigma_2, \ldots, \alpha_j, \sigma_{j+1} \rangle$, where $j \leq k$, a *state-terminated prefix*, or *st-prefix*, of p. Notice that an st-prefix is defined so that it always begins and ends in a state. Clearly, an st-prefix is a path in $\tau(AD)$. We also require a representation of the "opposite" of the outcome of interest to help define counterfactual paths. Given an outcome θ, we define *inverse outcome* $\overline{\theta}$ to be the set $\{\bar{l} \mid l \in \theta\}$. We now build upon these concepts to define counterfactual paths.

Definition 4
Given query $\mathscr{Q} = \langle v, AD, \theta \rangle$, a factual path f with respect to $\mathscr{Q}$, and inverse outcome $\overline{\theta}$, a *counterfactual path* is a path $f' = \langle \sigma_1, \alpha_1, \sigma_2, ..., \alpha_k, \sigma_{k+1} \rangle$ of $\tau(AD)$ satisfying the following conditions:

1. The factual path f and counterfactual path f' share an st-prefix $\rho = \langle \sigma_1, \alpha_1, \sigma_2, ..., \alpha_j, \sigma_{j+1} \rangle$ where $0 \leq j < k$.
2. $\overline{\theta} \subseteq \sigma_{k+1}$

In the following discussion, the set of all possible counterfactual paths with respect to the factual path f is denoted by $F' = \{f'_1, f'_2, ..., f'_{m'}\}$.

Condition 1 requires that any counterfactual path f' with respect to the factual path f shares an st-prefix ρ with f. Condition 2 requires that the fluent literals of the inverse outcome $\overline{\theta}$ final state are satisfied in the final state of f'.

In our example, it is straightforward to reason over AD_{soc} and obtain a counterfactual path f'_{soc} for f_{soc} that shares an st-prefix ρ_{soc} terminating in state σ_3 of f_{soc}. In both f_{soc} and f'_{soc}, Socrates is charged with impiety at state σ_2 and as a result is in prison in the successor state σ_3 because of law (4) in AD_{soc}. The path f'_{soc} shown in the Figure 1 diverges from f_{soc} at state σ_3, at which point event $e_3 = escapes(socrates, prison)$ occurs. Note that AD_{soc} contains no laws preventing

such an event from occurring. Escaping prison results in Socrates not being imprisoned in the successor state σ_4' of f_{soc}' as per law (5) in AD_{soc}. Because Socrates is not imprisoned in state σ_4', event $walksOutside(X)$ is executable and thus f_{soc}' is a valid path in which he avoids sentencing. Socrates is therefore alive in the final state σ_5'. It is easy to see that f_{soc}' satisfies the conditions of the definition of counterfactual paths. Figure 1 gives a graphical representation, with English descriptions of the events, of the divergence of f_{soc}' from f_{soc} with the occurrence of Socrates' counterfactual escape from prison. To simplify the presentation, we assume that f_{soc}' is the only counterfactual path for $\mathscr{Q}_{soc}$ and therefore $F_{soc}' = \{f_{soc}'\}$, although it is easy to see that in another of numerous possible counterfactual paths for f_{soc}, Socrates could have taken a walk instead of questioning the gods in the first state, avoiding death in the final state.

For every counterfactual path f' with respect to a factual path f, we can define a cause with context. CwCs capture information about the factual and counterfactual paths, the outcome of interest, and a possible cause. In the remainder of this section, we present the concepts required to represent CwCs using our framework. We begin by formally representing the divergence of a counterfactual path f' from a factual path f and use it to determine the event in f that represents a cause for the outcome of interest.

Definition 5
Let f and f' be a factual and counterfactual path, respectively, and let their maximal shared st-prefix be $\rho = \langle \sigma_1, \alpha_1, \sigma_2, \ldots, \alpha_j, \sigma_{j+1} \rangle$. We say that $j+1$ is the *divergence point* of f and f', denoted by $\delta(f, f')$.

Definition 6
Given a factual path f, a counterfactual path f', and a divergence point $\delta(f, f')$, the *cause* is the event $c = \alpha_{\delta(f,f')}$ of f.

Intuitively, event $\alpha'_{\delta(f,f')}$ of f' is the event that triggered the divergence from f towards a state in which the inverse outcome $\overline{\theta}$ is satisfied. Event $\alpha_{\delta(f,f')}$ is the event that "actually" happened, leading to the final state where the outcome of interest θ is satisfied. The representation of cause from Definition 6 follows the counterfactual definition of cause discussed earlier in the paper. In our example, we have already determined that f_{soc}' diverges from f_{soc} at the state σ_3, and so $\delta(f, f')_{soc} = 3$. In other words, if event $\alpha_{\delta(f,f')} = \alpha_3$ in the factual path f_{soc} had not occurred, ultimately leading to Socrates' death in σ_5, then event $\alpha'_{\delta(f,f')} = \alpha_3'$ of the counterfactual path f_{soc}' could have occurred instead, leading to an alternate sequence of events in which Socrates is alive in the final state of the sequence. Referencing event α_3 of f_{soc}, we can state that $c_{soc} = sentencedTo(socrates, death)$ is the cause of θ for this pair of f and f' as per Definition 6.

Definition 7
A *cause with context*, or *CwC*, for a query $\mathscr{Q} = \langle v, AD, \theta \rangle$ is the tuple $\mathscr{C} = \langle c, \theta, f, f' \rangle$ where c is a cause, θ is an outcome, f is a factual path, and f' is a counterfactual path.

The definition of *CwC* captures information about the relationship between a cause c and an

outcome θ by providing the paths needed to determine c and the query containing the information used to obtain the paths. In our example, the CwC

$$\mathscr{C}_{soc} = \langle sentencedTo(socrates, death), \neg isAlive(socrates), f_{soc}, f'_{soc} \rangle$$

represents a scenario in which the sentencing of Socrates is a possible cause of his death.

Computing Causes with Context

In the previous section, we presented a framework for formalizing CwCs. In this section, we demonstrate how it is possible to compute CwCs for a given query $\mathscr{Q}$ via Answer Set Programming (ASP) (Gelfond and Lifschitz 1988; Gelfond and Lifschitz 1991), a form of declarative programming that is useful in knowledge-intensive applications. In the ASP methodology, problem-solving tasks are reduced to computing answer sets of suitable logic programs. As such, ASP is well suited to the task of computing CwCs from event mentions and domain knowledge. We begin this section with a discussion of the syntax and semantics of Answer Set Programming.

Answer Set Programming. Let Σ be a signature containing constant, function and predicate symbols. Terms and atoms are formed as in first-order logic. A *literal* is an atom a or its strong negation $\neg a$. Literals are combined to form rules that represent both domain knowledge and event mentions in our approach. A *rule* in ASP is a statement of the form:

$$h \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$$

where h_i's (the head) and l_i's (the body) are literals and *not* is the so-called *default negation*. Intuitively, the meaning of default negation is the following: "if you believe $\{l_1, \ldots, l_m\}$ and have no reason to believe $\{l_{m+1}, \ldots, l_n\}$, then you must believe h". An ASP rule with an empty body is called a fact, and that in writing facts, the $\leftarrow$ connective is dropped. Rules of the form $\bot \leftarrow l_1, \ldots, \text{not } l_n$ are abbreviated $\leftarrow l_1, \ldots, \text{not } l_n$, and called *constraints*, intuitively meaning that $\{l_1, \ldots, \text{not } l_n\}$ must not be satisfied. A rule with variables (denoted by an uppercase initial) is interpreted as a shorthand for the set of rules obtained by replacing the variables with all possible variable-free terms. A *program* is a set of rules over Σ.

A consistent set S of domain literals is closed under a rule if $h \in S$ whenever $\{l_1, \ldots, l_m\} \subseteq S$ and $\{l_{m+1}, \ldots, l_n\} \cap S = \emptyset$. Set S is an answer set of a *not*-free program Π if S is the minimal set closed under its rules. The reduct, Π^S, of a program Π w.r.t. S is obtained from Π by removing every rule containing an expression "not l" s.t. $l \in S$ and by removing every other occurrence of not l. Finally, et S is an answer set of a program Π if S is the answer set of Π^S.

For a convenient representation of choices, in this paper we also use *constraint literals*, which are expressions of the form $m\{l_1, l_2, \ldots, l_k\}n$, where m, n are arithmetic expressions and l_i's are basic literals. A constraint literal is satisfied w.r.t. S whenever $m \leq |\{l_1, \ldots, l_k\} \cap S| \leq n$. Constraint literals are especially useful to reason about available choices. For example, a rule $1\{p, q, r\}1.$ intuitively states that exactly one of $\{p, q, r\}$ should occur in every answer set.

Translating a query $\mathscr{Q}$ to ASP. A query $\mathscr{Q} = \langle DD, \theta \rangle$ is translated to ASP as follows. First, the elements of $DD = \langle v, AD \rangle$ are translated into their ASP counterparts $DD_p = \langle v_p, AD_p \rangle$. For every event e_i in v, a fact $occurs(e, i)$ is included in v_p, indicating that event e occurred at step i of the story.

Set AD_p contains the ASP encoding of every dynamic causal law, state constraint, and executabilty condition of AD according to the mapping from Table 2. The encoding illustrated in the

Table 2: *ASP translations of action description laws.*

Law	ASP Representation
Dynamic Causal Law	$holds(l_0, I+1) \leftarrow holds(l_1, I), \ldots, holds(l_n, I), occurs(e, I).$
State Constraint	$holds(l_0, I) \leftarrow holds(l_1, I), \ldots, holds(l_n, I).$
Executability Condition	$\leftarrow holds(l_1, I), \ldots, holds(l_n, I), occurs(e, I).$

table follows a rather typical translation of $\mathscr{AL}$, see e.g. (Balduccini and Gelfond 2003). In this mapping, I is a variable ranging over all possible states, or time steps[3], in the evolution of the domain. Atom $holds(l, i)$ states[4] that a fluent literal l holds at time step i. For example, law (5) from AD_{soc} is translated to the following ASP rule:

$$\neg holds(inPrison(X), I) \leftarrow \ occurs(escapes(X, prison), I),\ holds(inPrison(X), I).$$

An ASP action description also contains rules formalizing the principle of inertia, which states that things generally stay as they are (McCarthy and Hayes 1969):

$$holds(F, I+1) \leftarrow fluent(F), holds(F, I), \text{not } \neg holds(F, I+1).$$

$$\neg holds(F, I+1) \leftarrow fluent(F), \neg holds(F, I), \text{not } holds(F, I+1).$$

Finally, θ is translated to ASP so that, for each fluent $f \in \theta$, the following constraint is added to θ_p:

$$\leftarrow \text{ not } holds(f, k+1).$$

where k is the number of events in the sequence v_p. Similarly, for every $\neg f \in \theta$, θ_p contains the constraint:

$$\leftarrow \text{ not } \neg holds(l, k+1).$$

In general, the translated query is represented by the set of rules $\Pi_{\mathscr{Q}} = v_p \cup AD_p \cup \theta_p$, containing rules corresponding to each element of the query.

Differentiating between Factual and Counterfactual Paths. Our approach to computing CwCs depends in part on the ability to differentiate between factual and counterfactual paths during computation. We facilitate this by enabling the reasoner to create labeled variants of the literals that encode a path by means of the following set of *typing rules*:

$$\Pi_T = \begin{cases} holds(L, I, T) \leftarrow holds(L, I), type(T). \\ \neg holds(L, I, T) \leftarrow \neg holds(L, I), type(T). \\ occurs(E, I, T) \leftarrow occurs(E, I), type(T). \end{cases}$$

Relation *type* specifies the type of path being considered by the reasoner. Its argument can be *fp*, for factual path, or *cfp*, for counterfactual path. The use of Π_T is illustrated next.

[3] Herein, the terms *state* and *time step* are used interchangeably.

[4] To ensure that all ASP literals are syntactically legal, if l is a negated fluent, $\neg f$, then we write the corresponding ASP literal as $\neg holds(f, i)$.

Computing Factual Paths from a Query. Given a program $\Pi_{\mathscr{Q}}$ representing the translated query $\mathscr{Q}$ and the set of typing rules Π_T, we define rules for computing factual paths as follows. Recall that the events of v_p are the events of the factual path as per Definition 3, each representing a transition between states in the factual path. Thus, in order to determine a factual path, we need to identify a possible initial state for it. The successive states in the path can then be inferred by means of AD_{soc_p}. In order to determine the initial state for a factual path, we use:

$$\Pi_{init} = \Big\{ 1\{holds(F,0), \neg holds(F,0)\}1 \leftarrow fluent(F).$$

The careful reader may notice that this rule is of form similar to the awareness axiom from (Gelfond and Kahl 2014). However, the intention behind the two axioms is arguably different.

We say that a factual path $f = \langle \sigma_1, e_1, \sigma_2, \ldots, e_k, \sigma_{k+1} \rangle$ is *encoded* by a set A of literals if-and-only-if $occurs(e_i, i, fp) \in A$ for every event e_i in f and $holds(l, i, fp) \in A$ for every state σ_i of f and $l \in \sigma_i$. Similarly, a counterfactual path is given by literals of the form $occurs(e_i, i, cfp)$ and $holds(l, i, cfp)$. Thus, the factual paths for a given query $\mathscr{Q}$ can be found by computing the answer sets of the program consisting of the translation $\Pi_{\mathscr{Q}}$ of $\mathscr{Q}$, the typing rules of Π_T, and the rules for computing the initial state of the factual path Π_{init}. More precisely:

Proposition 1.
Let F_p be the set of factual paths for a given query $\mathscr{Q}$. A path f_p belongs to F_p if-and-only-if f_p is encoded by some answer set of the program $\Pi_{P1} = \Pi_{\mathscr{Q}} \cup \Pi_{init} \cup \Pi_T \cup \{type(fp).\}$.

In the rest of the paper, Π_{f_i} denotes the set of literals of answer set A_i of Π_{P1} that encode a factual path. The set $F_p = \{\Pi_{f_1}, \Pi_{f_2}, \ldots, \Pi_{f_m}\}$ is the ASP representation of set F of all factual paths. We will use a set Π_{f_i} to aid the next program in the computation of counterfactual paths.

Computing Counterfactual Paths and Cause from Factual Paths. We can now define a program for computing the counterfactual paths for a factual path encoded by a set Π_{f_i} of literals from the previous computation, as well as the cause as per Definition 6. Each answer set of such program contains the elements of a CwC. As we did previously, we make use of the typing rules, this time specifying that the computed paths should be labeled as counterfactual paths by including the atom *type(cfp)*.

When computing counterfactual paths, we are looking for any paths in the transition diagram of the action description that share an st-prefix with the factual path in question and whose final state satisfies the inverse outcome. In this case, we cannot assume any initial knowledge of any of the events of (the non-prefix part of) the counterfactual path. For this reason, we begin by allowing considering all paths of the transition diagram of AD_p as candidate counterfactual paths using:

$$\Pi_G = \begin{cases} step(1..k). \\ 1\{occurs(E,I) : event(E)\}1 \leftarrow step(I). \end{cases}$$

The first rule specifies the number of transitions in a candidate path and the second rule states that any (single) event may occur at any step.

Recall Condition 1 from Definition 4, which requires any counterfactual path f' that diverges

from a factual path f to share an st-prefix ρ with f. This implies that f and f' must share the initial state:

Lemma 1
If f' is a counterfactual path for a factual path f, then the initial states of f' and f coincide.

Lemma 1 tells us that f and f' share an initial state, and it is straightforward to show that the theorem holds in the general case. Hence, the initial state of the counterfactual path can defined by:

$$\Pi_{\sigma_{init}} = \begin{cases} holds(L,1) \leftarrow holds(L,1,fp). \\ \neg holds(L,1) \leftarrow \neg holds(L,1,fp). \end{cases}$$

Intuitively, the purpose of $\Pi_{\sigma_{init}}$ is to produce literals of the form $holds(l,i)$ and $occurs(e,i)$, so that they can be used, together with the rules of AD_p, to reason about paths of $\tau(AD)$ stemming from the given initial state. Once desired paths have been found, the typing rules, together with a fact $type(cfp)$, will yield literals of the form $holds(l,i,cfp)$ and $occurs(e,i,cfp)$ to create the paths labeled *cfp*.

In order to satisfy Condition 1 of the definition of counterfactual path, we must ensure that any path found shares an st-prefix with the factual path being considered. This is accomplished via the following rules, which leverage the typed variants of relations *holds* and *occurs*:

$$\Pi_{div} = \begin{cases} 1\{div(I) : step(I)\}1. \\ \leftarrow holds(F,I,P1),\ \neg holds(F,I,P2),\ P1 \neq P2,\ div(\delta), I \leq \delta. \\ \leftarrow occurs(E,I,P1),\ \text{not } occurs(E,I,P1),\ P1 \neq P2,\ div(\delta),\ I < \delta. \end{cases}$$

These rules enable the reasoner to find counterfactual paths by "guessing" a possible divergence point and checking if the literals and events of the factual path (type *fp*) and of the counterfactual path (type *cfp*) are identical up to the point indicated by the guess.

Condition 2 of the definition of counterfactual path is enforced as follows. For each fluent $f \in \theta$, set $\overline{\theta}_p$ contains:

$$\leftarrow \text{ not } \neg holds(f,k+1).$$

Similarly, for every $\neg f \in \theta$, $\overline{\theta}_p$ includes:

$$\leftarrow \text{ not } holds(f,k+1).$$

which ensures that all computed counterfactual paths end in a state where $\overline{\theta}_p$ is satisfied.

Finally, we include a rule that computes the cause according to Definition 6 by identifying the event that occurs at the state indicated by the divergence point:

$$\Pi_{cause} = \Big\{ cause(E) \leftarrow div(I), occurs(E,I,f).$$

We can now find a set F'_p of counterfactual paths and corresponding causes for a factual path Π_{f_i} given by Π_{P1} above by computing the answer sets of the program consisting of the literals encoding the factual path Π_{f_i}, the translation of the action description AD_p, the inverse outcome $\overline{\theta}_p$, the rules for generating candidate paths Π_G, the rules for finding the divergence point of paths Π_{div}, the rule for isolating the cause in the factual path Π_{cause}, and the typing rules of Π_T. More precisely:

Proposition 2
Let F'_p be the set of counterfactual paths for a given factual path encoding Π_{f_i}. Given program $\Pi_{P2} = \Pi_{f_i} \cup AD_p \cup \overline{\theta}_p \cup \Pi_G \cup \Pi_{div} \cup \Pi_{cause} \cup \Pi_T \cup \{type(cfp).\}$, the following hold:

1. Path $f' \in F'_p$ if and only if f' is encoded by some answer set of Π_{P2}
2. Event e is a cause if and only if $cause(e)$ is in some answer set of Π_{P2}

In the following, $\Pi_{f'_i}$ denotes the set of literals from answer set A_i of Π_{P2} that encode a counterfactuals path. The set $F'_p = \{\Pi_{f'_1}, \Pi_{f'_2}, \ldots, \Pi_{f'_m}\}$ is the ASP representation of set F' of all counterfactual paths.

Summing up, given a query $\mathscr{Q} = \langle v, AD, \theta \rangle$, a factual path f for it can be found by computing an answer set of Π_{P1}; then, a counterfactual path f' for f can be found by computing an answer set of Π_{P2}. The same answer set also encodes cause e. Together with θ from $\mathscr{Q}$, this yields a CwC $\mathscr{C} = \langle e, \theta, f, f' \rangle$ with respect to $\mathscr{Q}$.

It is straightforward to translate the query from the Socrates example of the previous section into ASP using the approach given in this section. Then, using the programs given by Propositions 1 and 2 for computing factual paths, counterfactual paths, and causes, we can compute the CwC obtained in the previous section relating Socrates' death to his sentencing.

Related Work

To our knowledge, there are no related efforts to representing and reasoning about causes with context as we have presented here. The majority of research regarding counterfactuals and actual cause in artificial intelligence is concerned with modeling and evaluating provided counterfactual statements, e.g. *"If x had (or had not) been true, then y would be true"*, by simulating the proposed relationship in a model of causal dependencies and analyzing the behavior of system (Pearl 2009; Baral and Hunsaker 2007; Pereira and Saptawijaya 2016; Chockler and Halpern 2004). Our work differs from these approaches in that we infer the possible cause via counterfactual inference over a domain description and outcome of interest, rather than evaluating a statement that proposes a possible cause.

Conclusions and Future Work

In this paper we have introduced the notion of cause with context, or CwC, which is a variation on the concept of actual cause that provides us not only with a possible cause for an outcome of interest, but with additional information about how the cause fits into the world. We presented a novel framework linking CwCs and counterfactual reasoning and demonstrated it on a CwC for the cause of Socrates' death in the legal context from the introductory example. We have also presented an approach to computing CwCs via Answer Set Programming. Note that our approach to representing and reasoning about CwCs does not strictly rely upon the use of Answer Set Programming as we have shown here. It may be possible to formalize our framework using other well-known logical languages (e.g. Event or Situation Calculus) for reasoning about actions and change as long as they provide the ability to represent and reason about the occurrence of events and information about the state of the world before, during, and after them via a transition diagram.

The aim of the work presented here is to lay the foundations of CwCs from a representational standpoint, leveraging techniques from RAC and counterfactual reasoning to represent scenarios

and to identify possible actual causes for an outcome of interest. In future phases of our work we aim to conduct a case study to explore how our framework can handle common issues that arise when determining cause (e.g. overdetermination, preemption, contributory cause), investigate possible relationships between our work and existing approaches to evaluating causal relationships, explore how the contextual information in CwCs can be characterized (e.g. $\mathscr{C}_{soc}$ relates to the legal domain), and address issues of computational complexity.

References

BALDUCCINI, M. AND GELFOND, M. 2003. Diagnostic reasoning with a-prolog. *Theory and Practice of Logic Programming 3*, 4+ 5, 425–461.

BARAL, C. AND GELFOND, M. 2000. Reasoning Agents In Dynamic Domains. In *Workshop on Logic-Based Artificial Intelligence*. Kluwer Academic Publishers, 257–279.

BARAL, C. AND HUNSAKER, M. 2007. Using the probabilistic logic programming language p-log for causal and counterfactual reasoning and non-naive conditioning. In *IJCAI*. 243–249.

CHOCKLER, H. AND HALPERN, J. Y. 2004. Responsibility and blame: A structural-model approach. *Journal of Artificial Intelligence Research 22*, 93–115.

COLLINS, J. D. AND HALL, E. J. 2004. *Causation and counterfactuals*. MIT Press.

GELFOND, M. AND KAHL, Y. 2014. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents. The Answer-Set Programming Approach.* Cambridge University Press.

GELFOND, M. AND LIFSCHITZ, V. 1988. The stable model semantics for logic programming. In *Proceedings of ICLP-88*. 1070–1080.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing 9*, 365–385.

GELFOND, M. AND LIFSCHITZ, V. 1993. Representing Action and Change by Logic Programs. *Journal of Logic Programming 17*, 2–4, 301–321.

GINSBERG, M. L. 1986. Counterfactuals. *Artificial intelligence 30*, 1, 35–79.

HALPERN, J. Y. AND HITCHCOCK, C. 2011. Actual causation and the art of modeling. *arXiv preprint arXiv:1106.2652*.

HAUGH, B. A. 1987. Simple causal minimizations for temporal persistence and projection. In *AAAI*. 218–223.

LEWIS, D. 1973. Counterfactuals and comparative possibility. *Journal of Philosophical Logic 2*, 4, 418–446.

LIFSCHITZ, V. 1987. Formal theories of action (preliminary report). In *IJCAI*. 966–972.

LIN, F. 1996. Embracing causality in specifying the indeterminate effects of actions. In *Proceedings of the thirteenth national conference on Artificial intelligence-Volume 1*. AAAI Press, 670–676.

MCCAIN, N., TURNER, H., ET AL. 1997. Causal theories of action and change. In *AAAI/IAAI*. 460–465.

MCCARTHY, J. AND HAYES, P. J. 1969. Some philosophical problems from the standpoint of artificial intelligence. *Readings in artificial intelligence*, 431–450.

MERCK, C. A. AND KLEINBERG, S. 2016. Causal explanation under indeterminism: A sampling approach. In *AAAI*. 1037–1043.

PEARL, J. 2009. *Causality*. Cambridge University Press.

PEREIRA, L. M., APARÍCIO, J. N., ALFERES, J. J., AND JOAQUIM, P. 1991. Counterfactual reasoning based on revising assumptions.

PEREIRA, L. M. AND SAPTAWIJAYA, A. 2016. Counterfactuals, logic programming and agent morality. *Logic, Argumentation & Reasoning. Berlin: Springer (forthcoming)*.

THIELSCHER, M. 1997. Ramification and causality. *Artificial intelligence 89*, 1-2, 317–364.

VENNEKENS, J., BRUYNOOGHE, M., AND DENECKER, M. 2010. Embracing events in causal modelling: Interventions and counterfactuals in cp-logic. In *European Workshop on Logics in Artificial Intelligence*. Springer, 313–325.