Explaining Actual Causation via Reasoning about Actions and Change

A Thesis

Submitted to the Faculty

of

Drexel University

by

Emily C. LeBlanc

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

June 2019

Acknowledgments

I am profoundly grateful to my thesis advisor Marcello Balduccini for his invaluable feedback, guidance, and support while pursuing my PhD. I am very thankful for all he has taught me and for his encouragement and availability over the last six years. Without him, none of this would have been possible. His influence on my work and my development as a researcher and a professional has been substantial. I am truly fortunate to have Marcello to look up to.

I would also like to thank Santiago Ontañón for taking on the role of my official advisor for the last few years of my time in the program. His support was crucial to my ability to complete this work, and I thank him for bringing me into the GAIMS lab and providing a perfect environment in which to work.

I am very grateful to my entire dissertation committee, Santiago Ontañón, Marcello Balduccini, Dario Salvucci, Vasilis Gkatzelis, Steven Gustafson, and Luis Tari, for their thoughtful questions and feedback which substantially improved the quality of this dissertation.

I am very thankful for the constant support of the Drexel Department of Computer Science that I've received over the course of my time in this program. The opportunities provided to me by the department have significantly contributed to my development as a professional in my field. I am also thankful to the members of the GAIMS lab, who made the months leading up to the completion of my dissertation fun and interesting. I have been very happy to be a part of this group.

I am also very grateful to Bill Regli for bringing me into the department as a PhD student in 2013, and for involving me from that start in a fascinating project that gave me unparalleled experience and ultimately set the stage for finding my research interests.

I am thankful to the many researchers in the community with which I have had numerous conversations about this work, namely, Joost Vennekens for providing invaluable feedback and ongoing guidance, as well as Chitta Baral for his mentorship at the Doctoral Consortium of the 15th International Conference on Principles of Knowledge Representation and Reasoning, whose

thoughtful advice resulted in a sharp focusing of my dissertation topic.

I also want to thank the researchers I've worked with during two internships with GE Global Research, Luis Tari and Steven Gustafson, for allowing to me to work on new and interesting projects. These experiences enabled me to work on real-world problems with talented industry professionals, and greatly encouraged the development of my independence as a researcher.

I am very grateful to my mother, father, and sister for their encouragement and support throughout my life and over the course of my education. Each of them has motivated my creativity and curiosity from childhood to present day. I would not be here without them, and I am always appreciative of the positive influences they have on me.

I also want to thank Chris, Meagan, Catherine, and Ulysses Macklin for their friendship over many years. They infuse my life with humor and fun, and their support has been and continues to be invaluable to me. They inspire me in many ways and I am grateful that they are in my family.

Finally, I want express my gratitude for my partner, Duc Nguyen. Detailing the countless ways that he positively influences my life would likely result in another dissertation. His understanding, patience, and love have been instrumental in helping me get through the hard times and celebrate the good ones. I am incredibly lucky to have him in my life and I am grateful for every day together.

Table of Contents

List of Tables

List of Figures

Abstract

Explaining Actual Causation via Reasoning about Actions and Change
Emily C. LeBlanc
Santiago Ontañón, Ph.D. and Marcello Balduccini, Ph.D.

The goal of this research is to investigate and demonstrate the suitability of action languages and answer set programming (ASP) to design and realize a novel framework for explaining *actual causation*. Actual causation is a broad term that encompasses all possible antecedents that have played a meaningful role in producing a consequence. Attempts to characterize reasoning about actual causation have largely pursued counterfactual analysis of a scenario, inspired by the intuition that if X caused Y, then not Y if not X. However, it has been widely documented that the counterfactual criteria alone is problematic and fails to recognize causation in a number of straightforward cases. Departing from a counterfactual reasoning approach, our framework favors reasoning about the underlying causal mechanisms of the scenario itself in order to explain how an outcome of interest came to be. The framework leverages techniques from Reasoning about Actions and Change to support reasoning about domains that change over time in response to a sequence of events. The action language $\mathcal{AL}$ enables us to represent a scenario in terms of the evolution of the state of the world over the course of the scenario's events. $\mathcal{AL}$ lends itself naturally to an automated translation in Answer Set Programming (ASP), using which, reasoning tasks of considerable complexity can be specified and executed. In this dissertation, we present a theoretical framework for reasoning about actual causation and demonstrate that the framework enables reasoning about traditionally challenging examples of actual cause. We also present a sound and complete implementation of the theoretical framework in ASP, along with a collection of empirical studies that evaluate and analyze the framework's performance on a number of novel and challenging problems.

Chapter 1: Introduction

"The complexities of cause and effect defy analysis."

— Douglas Adams

The goal of this research is to investigate and demonstrate the suitability of action languages and answer set programming (ASP) to design and realize a novel framework for reasoning about and explaining *actual causation*. Also referred to as causation in fact, *actual cause* is a broad term that encompasses all possible antecedents that have played a meaningful role in producing a consequence [1]. Reasoning about actual cause concerns determining how a particular consequence came to be in a given scenario, and the topic been studied extensively in numerous fields including philosophy, law, and, more recently, computer science and artificial intelligence.

Attempts to mathematically characterize actual causation have largely pursued counterfactual analysis of structural equations (e.g., [2–6]), neuron diagrams [7, 8], and other logical formalisms (see e.g., [9]). Counterfactual accounts of actual causation are inspired by the human intuition that if X caused Y, then not Y if not X [10]. At a high level, this approach involves looking for *possible worlds* in which Y is true and X is not. If such a world is found, then X is not a cause of Y. It has been widely documented, however, that the counterfactual criteria alone is problematic and fails to recognize causation in a number of common cases such as overdetermination (i.e., multiple causes for the effect), preemption (i.e., one cause "blocks" another's effect), and contributory causation (i.e., causes must occur together to achieve the effect) [11, 12]. Subsequent approaches have addressed some of the shortcomings associated with the counterfactual criterion by modifying the existing definitions [13, 14], introducing supplemental definitions [9, 15, 16], and by modeling time [17] with some improved results. In spite of these improvements, there is still no widely accepted definition of actual cause.

Departing from the counterfactual intuition and reasoning about possible worlds, our framework favors reasoning about the underlying causal mechanisms of the scenario itself in order to

explain actual causation of an outcome of interest. Our framework uses techniques from Reasoning about Actions and Change (RAC) to support reasoning about domains that change over time in response to a sequence of events. The action language $\mathcal{AL}$ [18] enables us to represent a scenario as the evolution of state over the course of the scenario's events. Moreover, the elements of $\mathcal{AL}$ semantics can be used to define notions of direct and indirect cause, and the language's solution to the frame problem can be leveraged to detect the "appearance" of an outcome of interest in a scenario. Our position on reasoning about actual cause is supported by recent work in the area [19, 20] that shares similar intuition to our own about the appearance of outcomes. Finally, $\mathcal{AL}$ lends itself naturally to an automated translation in ASP, using which, reasoning tasks of considerable complexity can be specified and executed.

1.1 Explaining Actual Causation

Sophisticated actual causal reasoning has long been prevalent in human society and continues to have an undeniable impact on the advancement of science, technology, medicine, and other fields that are critical to the success of modern society. From the development of ancient tools to modern root cause analysis in business and industry, reasoning about causal influence in a historical sequence of events enables us to diagnose the cause of an outcome of interest and gives us insight into how to bring about, or even prevent, similar outcomes in future scenarios. Consider problems such as explaining the occurrence of a set of suspicious observations in a network security system, reasoning about the efficiency of actions taken in an emergency evacuation scenario, or investigating how an automatically generated workflow produced some unexpected results. It is easy to imagine that analyzing such (potentially very complex) scenarios requires the ability to represent and reason about how the state of the world has been changed by the scenario's events to produce some outcome of interest.

Consider the behavior of an advanced cyber-physical system such as a self-driving car, reasoning about causation (e.g., blame or praise) becomes significantly more complex – the car likely contains a large number of software and hardware modules (possibly from different vendors), there may be other cars and pedestrians involved in the scenario of interest, and there may have been

wireless communication with other vehicles or a central server, all of which may influence the actions taken by the car's control module over the course of its drive. To reach an intuitively satisfactory explanation of why some outcome of interest came to be in such a domain, the insights that have been produced by the decades-long study of actual causation seem indispensable.

Modern work on actual causation originated in philosophy with the seminal paper by Lewis [10]. His work, like that of other philosophers following him, was primarily theoretical and not intended to be put to practical use. The famous Halpern-Pearl (HP) paper [3] initiated interest in this concept within the field of AI and it constitutes a first milestone on the way towards applications of the concept of actual causation. However, neither the HP paper nor the many that have followed it (e.g., see Chapter 7) have yet reached the point where their results could be directly applied, for example, in the context of a self-driving car as proposed here, or in other similarly complex scenarios. We believe that this is due, at least in part, to a lack of distinction between the laws that govern individual states of the world and events whose occurrence cause state to evolve.

We also believe that our approach to reasoning about *what actually happened* rather than *what could have happened* sets us apart from the majority of the work in this research area (discussed in greater detail in Chapter 7), and our choice of $\mathcal{AL}$ as a formalism positions the approach for potential use in practical settings via translation to an implementation in ASP.

The primary contributions of this dissertation are as follows:

1. A novel theoretical framework for reasoning about actual causation in terms of the semantics of the action language $\mathcal{AL}$.

2. A sound and complete implementation of the framework in ASP.

3. Empirical studies of the practical feasibility of the implementation on increasingly large and complex novel causal scenarios, with respect to time.

1.2 Organization

We now present the organization of the dissertation.

- Chapter 2 contains background information about the environments of interest for which the framework is defined, the action language $\mathcal{AL}$, and ASP.

- Chapter 3 presents the theoretical framework for explaining actual causation. The chapter contains a novel running example which we use to aid our discussion of the framework's definitions. In addition to presenting the definition of direct cause, we present a simple notion of indirect causation and identify two important drawbacks to the definition. Next, we present an improved definition that addresses the identified shortcomings and provides additional information about indirect causes. Finally, we draw initial conclusions about the framework's ability to handle traditionally challenging cases of causation.

- Chapter 4, we use the framework to reason about examples from the literature that have been used to challenge the counterfactual definition of actual cause, as well as a novel example inspired by the self-driving car scenario outlined above.

- Chapter 5 provides implementations of the theoretical framework in ASP and presents theoretical results about soundness and completeness of the program for computing direct cause and the improved definition of indirect cause. We also present ASP translations of a subset of the examples from Chapter 4.

- Chapter 6 presents experimental results from empirical studies aiming to assess the practical feasibility of the approach with respect to time. To the best of our knowledge, there is no established set of benchmarks for the type of reasoning presented in this dissertation, and so we have generated a set of novel problem instances that allow us to examine and make initial conclusions about the performance of the implementation on a number of interesting and increasingly complex causal scenarios.

- Chapter 7 provides in-depth discussions about related approaches. We will present an overview of technical approaches and comparative discussion for the most well-known and widely studied approach in the field [3], the work that led us to our intuition about representing scenarios as the evolution of state in response to events [21], and the work that leverages

similar intuition to our own in reasoning about scenarios [19].

- Chapter 8 presents our conclusions and suggests avenues for future work.

Chapter 2: Background

Background

This chapter contains background information on the classes of problems for which the framework is designed, action language $\mathcal{AL}$, which we use to represent domains that change over time, and answer set programming which we use to implement the computation of direct and indirect causes.

2.1 Environments of Interest

In this work, we use the term *dynamic domain* to refer to an environment whose state changes in response to the execution of actions. Dynamic domains of interest in this work satisfy the following conditions:

1. state evolves in discrete steps
2. states are defined by a set of boolean statements called *fluents*
3. the effects of events are instantaneous (i.e., there are no time-delayed effects)
4. the effects of events are deterministic
5. all fluents are observable, that is to say there is no uncertainty in the values of fluents

These conditions reduce the complexity of the presentation and allow the reader to focus on the core contributions of the work, but our study leads us to speculate that likely none is essential to the validity of the study. In fact, we believe that most can be relaxed in future studies by adopting existing approaches from the literature. For instance, boolean statements could be replaced by numerical values as in $\mathcal{C}+$[22], actions and events with duration could be dealt with by introducing processes [23, 24], non-determinism could be introduced in a way similar to [25, 26], and dealing with non-observable fluents could reflect the contributions of [27].

2.2 Reasoning about Actions and Change

Reasoning about Actions and Change is concerned with representing the properties of actions [28]. 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 [29–34]. In our work, we aim to leverage mentions of observed events and domain knowledge to determine a detailed picture of a scenario over time. Here we provide a more technical discussion of RAC and action language $\mathcal{AL}$.

Action languages [35] are formalisms used to describe the effects of actions, or events[1], in a domain. In these languages, the domain is represented by a transition diagram, e.g. a directed graph with nodes corresponding to the states of the domain and arcs corresponding to transitions between states. Arcs are labeled by events, according to the intuition that transitions between states are initiated by the occurrence of events.

Action languages are good candidates for the high-level specification of domain models as their relative simplicity and clear intuitive reading of the statements usually allow the designer to have reasonable confidence in the correctness of their domain specifications. For the representation of domains and their evolution over time, in this work we rely on the action language $\mathcal{AL}$ [18], whose syntax and semantics we present next.

2.2.1 Syntax of $\mathcal{AL}$

The language $\mathcal{AL}$ builds upon an alphabet consisting of a set $\mathcal{F}$ of *fluents* and a set $\mathcal{E}$ of *elementary events*. Fluents are boolean properties of the domain, whose truth value may change over time. A *(fluent) literal* is a fluent f or its negation $\neg f$. Additionally, we define the complement $\overline{f} = \neg f$ and $\overline{\neg f} = f$. A single elementary event is denoted by its element e of $\mathcal{E}$. A *compound event* is a set of elementary events $\epsilon = \{e_1, \ldots, e_n\}$. A state σ is a set of literals, the properties of which we present later in this discussion. If $f \in \sigma$, we say that f *holds* in σ. A signature is a tuple $\Phi = \langle \mathcal{F}, \mathcal{E} \rangle$, whose components are defined above. Given a signature, the laws of $\mathcal{AL}$ are as follows. A statement of

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

the form

$$d : e \textbf{ causes } l_0 \textbf{ if } l_1, \ldots, l_n \tag{2.1}$$

is called a *dynamic (causal) law*, where d is a constant used to name the law. Intuitively, a law d of form (2.1) says that if elementary event e occurs in a state where literals $l_1, \ldots, l_n$ hold, then literal l_0 will hold in the next state. Note that the name of the law is not used to define the semantics of the language, and will thus be omitted to simplify the presentation when possible[2]. Next, a statement

$$s : l_0 \textbf{ if } l_1, \ldots, l_n \tag{2.2}$$

is called a *state constraint*, where s is the name of the law, and says that in any state in which literals $l_1, \ldots, l_n$ hold, l_0 also holds. We say that l_0 is the *consequence* of the law s and we often refer to the consequence of a state constraint s as $consq(s)$. It will be useful later to refer to the set of literals $\{l_1, \ldots, l_n\}$ as $prec(s)$. A statement of form (2.2) allows for an elegant and concise representation of *indirect effects*, or *ramifications*, of events which enhances the expressive power of the language. Finally, a statement of the form

$$\iota : e \textbf{ impossible\_if } l_1, \ldots, l_n \tag{2.3}$$

is called an *executability condition*, where ι is the name of the law, and states that an elementary event e cannot occur when $l_1, \ldots, l_n$ hold. A set of statements of $\mathcal{AL}$ is called an *action description*.

2.2.2 Semantics of $\mathcal{AL}$

A set S of literals is *closed under a state constraint* (2.2) if $\{l_1, \ldots, l_n\} \not\subseteq S$ or $l_0 \in S$. Set S is *consistent* if, for every $f \in \mathcal{F}$, 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* σ of an action description AD is a complete and consistent set of fluent literals closed under the state constraints of AD.

[2] In the chapter on the encoding of $\mathcal{AL}$ to ASP, we will assume that a name has been specified for each law.

Given an elementary 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 (2.1) such that $\{l_1, \ldots, l_n\} \subseteq \sigma$. Given a compound event $\epsilon = \{e_1, \ldots, e_n\}$, the set of direct effects of ϵ in σ is denoted by $E(\epsilon, \sigma) = E(e_1, \sigma) \cup \ldots \cup E(e_n, \sigma)$. 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 every state constraint in Z. Finally, an event e is *non-executable* in a state σ if there exists an executability condition (2.3) such that $\{l_1, \ldots, l_n\} \subseteq \sigma$. Otherwise, the event is *executable*[3] in σ.

The semantics of an $\mathcal{AL}$ action description AD is defined by its *transition diagram* $\tau(AD)$, a directed graph $\langle N, A \rangle$ such that N is the collection of all states of AD, A is the set of all triples $\langle \sigma, \epsilon, \sigma' \rangle$ where σ, σ' are states, ϵ is an event executable in σ, and σ' satisfies the *successor state equation*:

$$\sigma' = Cn_Z(E(\epsilon, \sigma) \cup (\sigma \cap \sigma')) \tag{2.4}$$

where Z is the set of all state constraints of AD. The argument of Cn_Z in (2.4) is the union of the set of direct effects $E(e, \sigma)$ for all $e \in \epsilon$ with the set $\sigma \cap \sigma'$ of the literals *preserved by inertia*. The application of Cn_Z adds the indirect effects to this union. A triple $\langle \sigma, \epsilon, \sigma' \rangle \in E$ is called a *transition* of $\tau(AD)$ and σ' is a *successor state of σ* (under ϵ). A sequence $\langle \sigma_1, \epsilon_1, \sigma_2, \ldots, \epsilon_k, \sigma_{k+1} \rangle$ is a *path of* $\tau(AD)$ of length k if every $\langle \sigma_i, \epsilon_i, \sigma_{i+1} \rangle$, $1 \leq i \leq k$, is a transition in $\tau(AD)$. We denote the *initial state* of a path ρ by σ_1. We say that states σ_i and $\sigma_{i'}$ are *sequential* if $i' = i + 1$.

An action description AD has *emergent non-deterministic behavior* if, for some ϵ and σ there exist multiple σ' such that (2.4) is satisfied[36]. In this dissertation, we focus on action descriptions without emergent non-deterministic behavior[4].

The definition of $\tau(AD)$ is based upon [37] and is the product of an intensive investigation into the nature of causality (see also [34, 38]). Arriving at this definition required a deep understanding of the nature of causal effects of actions in the presence of complex interrelations between fluents. An additional level of complexity is introduced by the need to specify what is *not* changed by

[3]Note that an event may occur without having an effect on the state of the world, commonly referred to in the literature as a NOP action.

[4]Action description $\{q$ **if** $\neg r, p; r$ **if** $\neg q, p; a$ **causes** $p\}$ has an emergent non-deterministic behavior.

actions, a well-known challenge in AI called the *frame problem*. The challenge is often reduced to the problem of finding a concise and accurate representation of the *inertia axiom* – a default which says that *things normally stay as they are* [39]. The search for such a representation substantially influenced AI research during the last twenty years. An interesting account of history of this research together with some possible solutions can be found in [40].

2.3 Answer Set Programming

Answer Set Programming [41, 42] is 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 demonstrated by a substantial body of literature (see, e.g., [43] for the most closely related of these works), $\mathcal{AL}$ lends itself quite naturally to an automated translation to Answer Set Programming [41, 42], using which, reasoning tasks of considerable complexity can be specified and executed (see, e.g., [44–46]). As such, ASP is well suited to the task of computing actual causes and is our choice of language for the implementation of the $\mathcal{AL}$ framework presented in Chapter (Framework).

2.3.1 Syntax of ASP

The syntax of ASP is determined by a signature Σ consisting of types, typed object constants, as well as typed function and predicate constants. We will assume that the signature contains symbols for integers and for the standard relations of arithmetic. Terms are built as in typed first-order languages.

Atoms are expressions of the form $p(t_1, \ldots, t_n)$, where p is a predicate symbol of arity n and t's are terms of suitable types. Literals are atoms or negated atoms of the form $\neg p(t_1, \ldots, t_n)$. The symbol $\neg$ is called *classical* or *strong* negation. In our further discussion we often write $p(t_1, \ldots, t_n)$ as $p(\bar{t})$. Literals of the form $p(\bar{t})$ and $\neg p(\bar{t})$ are called complementary. By $\bar{l}$ we denote a literal complementary to l. Literals and terms not containing variables are called *ground*.

For convenient representation of cardinality, we additionally use *aggregate literals* from ASP-

Core-2 [47]. The syntax of an *aggregate element* are as follows

$$t_1, \ldots, t_m : l_1, \ldots, l_n$$

where $t_1, \ldots, t_m$ are terms $l_1, \ldots, l_n$ are literals of form l or not l for $m \geq 0$ and $n \geq 0$. An *aggregate atom* has the form:

$$\#count\{e_1; \ldots; e_n\} = u$$

where $e_1; \ldots; e_n$ are aggregate elements for $n \geq 0$ and $\#count$ is an *aggregate function name*[5]. Given an aggregate atom a, the expressions a and *not* a are *aggregate literals*.

A *rule* in ASP is a statement of the form:

$$h \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n \tag{2.5}$$

where $n \geq 1$, h (the *head*) and l_i's (the *body*) are literals and aggregate 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". For a given rule r, we refer to the head h as $head(r)$, the set $\{l_1, \ldots, l_m\}$ as $pos(r)$, and the set $\{l_1, \ldots, l_m\}$ as $neg(r)$. An ASP rule with an empty body is called a fact. 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 pair $\langle \Sigma, \Pi \rangle$ where Σ is a signature and Π is a collection of rules over Σ is called a *logic program*, or simply *program*. We often denote such a pair by its second element Π, and the corresponding signature is in turn denoted by $\Sigma(\Pi)$. The sets of all ground terms, atoms, and literals over Σ are

[5]See [47] for a full listing of aggregate function names and their semantics.

denoted by $terms(\Sigma)$, $atoms(\Sigma)$, and $lit(\Sigma)$, respectively. Consistent sets of ground literals over signature Σ, containing all arithmetic literals which are true under the standard interpretation of their symbols, are called *states* of Σ and are denoted by $states(\Sigma)$.We say that a literal $l \in lit(\Sigma)$ is *true* in a state X of Σ if $l \in X$ and that l is *false* in X if $\bar{l} \in X$. The satisfaction of a aggregate literal is based on the evaluation of the literal w.r.t the literal's aggregate function. The approach is described in [47], and we present the semantics of the aggregate function $\#count$ in the next section.

Grounding a program Π refers to applying to each rule $r \in \Pi$ all possible substitutions from the variables in r to the set of constants in of signature Σ. For grounding to be possible, all rules of the program should be *safe*, that is to say that all variables that appear in a rule r must be a member of $pos(r)$. Rules that do not satisfy this condition are *unsafe*. The satisfaction of aggregate literals is determined by this mapping and is described in detail in[47]. A *ground instantiation* of a program Π is denoted by $ground(\Pi)$. The terms, atoms, and literals of a program Π are denoted respectively by $terms(\Pi)$, $atoms(\Pi)$, and $literals(\Pi)$.

2.3.2 Semantics of ASP

The answer set semantics of a logic program Π assigns Π to a collection of *answer sets*– consistent sets of ground literals over signature $\Sigma(\Pi)$. Answer sets correspond to beliefs which can be built by a rational reasoner on the basis of the rules of Π. In the construction of these beliefs, the reasoner aims to satisfy the rules of Π, understood as constraints of the form "if you believe in the body of the rule, you must also believe in the head". Finally, they should adhere to the *rationality principle* which says that one shall not believe anything they are not forced to believe.

An *aggregate function* is a mapping from sets of tuples of terms to terms. The aggregate function associated with the $\#count$ function name maps a set T of tuples of ground terms to a ground term as follows:

$$\#count(T) = |T| \tag{2.6}$$

Answer sets are defined for programs whose rules do not contain default negation, also referred to as *not*-free programs. Let Π be a *not*-free program and let X be a consistent set of literals. We say that X is *closed* under Π if for every rule $head \leftarrow body$ of Π, $head$ is true in X whenever body is true in X. For a constraint, this condition means that $body$ is not contained in X.

Definition 1. *(Answer set of a* not*-free program). A consistent set of literals, X, is an* answer set of a program Π not containing default negation *if X is closed under all rules of Π and X is set-theoretically minimal among the sets satisfying the first property.*

It can be shown that a program without default negation can have at most one answer set. Extending the definition to an arbitrary program Π (which may contain default negation), computing the answer sets of Π is reduced to the computation of answer sets of programs without default negation. We say that the *reduct* Π^X of Π relative to X is the set of rules $l_0 \leftarrow l_1, \ldots, l_m$ for all rules of form (2.5) in Π such that $l_{m+1}, \ldots, l_n \notin X$. Formally,

Definition 2. *(Reduct of an arbitrary program). Let Π be an ASP program. For any set X of literals, let Π^X be the program obtained from Π by removing:*

- *each rule $r \in \Pi$ such that $neg(r) \cap X \neq \emptyset$*
- *all expressions of the form not l in the bodies of the remaining rules.*

It is easy to see that Π^X is a *not*-free program. We can now define the notion of answer set of an arbitrary program.

Definition 3. *(Answer set of a basic ASP program). A consistent set of literals, X, is an* answer set of a basic ASP program Π *if it is an answer set of Π^X.*

2.3.3 Properties of Answer Sets

The following properties of answer sets will be useful in Chapter 5.

Proposition 1. *(Supportedness.) Given an answer set X of Π, for every literal $a \in X$ there is some rule r from $ground(\Pi)$ whose body is satisfied in X and whose head contains a.*

It is easy to see that this is trivially true for facts of X. A related property of answer sets is that of *closedness*, which guarantees that whenever the body of a rule r is satisfied in X, then the head of r is a member of A. With the background of the work established, we proceed to the presentation of the main contribution of the dissertation.

Chapter 3: Theoretical Framework

In this section, we present the causal reasoning framework alongside a novel example that showcases the ability of the framework to reason about the direct and indirect causal influence of *events* over *literals*, originally presented in [48]. We will then present and discuss two counterexamples to our original definition of indirect cause. Following that, we will present an improved definition and show that it is able to solve the counterexamples. Finally, we will discuss some technical and conceptual advantages of the improved definition of indirect cause over the original definition.

Given a consistent set of literals θ representing an outcome of interest, an action description AD describing the effects of events in a dynamic domain, and a path ρ in $\tau(AD)$ corresponding to a scenario of interest, we construct a *problem* $\psi = \langle \theta, \rho, AD \rangle$. The framework can be used to reason over the elements of a problem ψ to identify events that are responsible for causing the literals of θ to hold simultaneously in a state of ρ. We now present the details of the running example.

3.1 Running Example

Let $\psi_E = \langle \theta_E, \rho_E, AD_E \rangle$ be a problem representing the running example. The outcome of interest in our example is $\theta_E = \{A, B, C, D, E, F\}$. The following action description AD_E characterizes elementary events in the example domain:

$$e_1 \textbf{ causes } A \textbf{ if } \neg A \tag{3.1}$$

$$e_2 \textbf{ causes } A \textbf{ if } \neg A \tag{3.2}$$

$$e_3 \textbf{ causes } C \textbf{ if } \neg C \tag{3.3}$$

$$e_4 \textbf{ causes } E \textbf{ if } \neg E \tag{3.4}$$

$$e_5 \textbf{ causes } F \textbf{ if } \neg F \tag{3.5}$$

$$e_5 \textbf{ causes } C \textbf{ if } \neg C \tag{3.6}$$

$$B \textbf{ if } C \tag{3.7}$$

$$D \textbf{ if } E, F \tag{3.8}$$

The dynamic law (3.1) states that e_1 will cause A to hold if it does not already hold when the event occurs. Similarly, laws (3.2) through (3.6) describe the direct effects of events e_2, e_3, e_4, and e_5. The state constraint (3.7) tells us that B holds whenever C holds, and the state constraint (3.8) tells us that D holds whenever both E and F hold. Note that although there are no executability conditions in this action description, it is straightforward to use them to model the events in AD_E in greater detail.

Table 3.1: Tabular representation of path $\rho_E \in \tau(AD_E)$.

State	*Event*
$\sigma_1 = \{\neg A, \neg B, \neg C, \neg D, \neg E, \neg F\}$	$\epsilon_1 = \{e_1, e_2\}$
$\sigma_2 = \{A, \neg B, \neg C, \neg D, \neg E, \neg F\}$	$\epsilon_2 = \{e_3\}$
$\sigma_3 = \{A, B, C, \neg D, \neg E, \neg F\}$	$\epsilon_3 = \{e_4, e_5\}$
$\sigma_4 = \{A,\ B,\ C,\ D,\ E,\ F\}$	–

The dynamics of the scenario are given by path $\rho \in \tau(AD_E)$. Path ρ consists of three compound events, $\epsilon_1 = \{e_1, e_2\}$, $\epsilon_2 = \{e_3\}$, and $\epsilon_3 = \{e_4, e_5\}$. Table 3.1 shows the evolution of state in ρ_E in response to these events. The first column lists each state of ρ_E, and the second column gives the event α_i that caused a transition to the subsequent state. The outcome θ_E is not satisfied in the first three states of the path, however, the events of ρ_E have somehow caused the outcome to be satisfied in state σ_4.

By examining the laws of AD_E together with the states and transitions of ρ_E, the reader can reason about which event(s) directly and (or) indirectly caused every literal in θ_E to hold by state σ_4. In the first transition, for instance, we see that events e_1 and e_2 overdetermining direct causes of A holding in state σ_2 according to laws (3.1) and (3.2), respectively. Next, event e_3 directly causes C to hold in state σ_3 according to law (3.3). In the same transition, e_3 indirectly causes B because of laws (3.3) and (3.7). Here, the direct effect of e_3 occurring in σ_2 was needed to satisfy the state

Table 3.2: Explanation of how each literal of θ_E was caused in path ρ_E.

Literal	*Compound Event*	*Direct Cause*	*Indirect Cause*	*Laws*
A	$\epsilon_1 = \{e_1, e_2\}$	e_1	–	(3.1)
		e_2	–	(3.2)
B	$\epsilon_2 = \{e_3\}$	–	e_3	(3.3),(3.7)
C		e_3	–	(3.3)
D	$\epsilon_3 = \{e_4, e_5\}$	–	e_4, e_5	(3.4),(3.5),(3.8)
E		e_4	–	(3.4)
F		e_5	–	(3.5)

constraint (3.7), which in turn caused B to hold. In the final transition, e_4 and e_5 directly cause E and F, respectively, as per (3.4) and (3.5). Finally, the co-occurrence of e_4 and e_5 in this transition indirectly causes D in accordance with laws (3.4), (3.5), and (3.8). This is a case of contributory cause in which the direct effects of e_4 an e_5 were both required to satisfy the preconditions of the state constraint (3.8), which results in D holding in state σ_4. Moreover, notice that if e_3 had not occurred in σ_2, then e_5 would have caused C to hold by law (3.6). However, we do not identify e_5 as a cause because it was preempted by e_3.

Table 3.2 summarizes the results of our reasoning over the problem. Each row of the table (or sets of rows in cases of overdetermination) characterizes the causation of each literal $l \in \theta_E$ by row. The first column lists the literal l in θ_E that is being explained. The second, third, and fourth columns tell us which event(s) caused l to hold, either directly or indirectly. As a reference for the reader, the final column specifies the laws of AD_E that are relevant to the causation of each l.

In this example, we needed only to reason over the laws of AD_E and the path ρ_E, to produce a fine-grained causal explanation about the direct and indirect causation of the literals of θ_E. We were also able to accurately identify causation in cases of overdetermination, contributory cause, and preemption. The goal of the theoretical framework is to mathematically characterize the type of reasoning process that we used to mentally solve this running example.

3.2 Framework Definitions

Here we present the definitions of the framework, and use them to characterize direct and indirect causation of every literal in θ_E in path ρ_E for our example, as in Table 3.2.

3.2.1 Transition States and Causing Compound Events

The first step in explaining how an outcome θ came to be in path ρ is to identify a *transition state* of the outcome in ρ. A transition state tells us when the outcome of interest *appears* in the path.

Definition 4. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a state σ_j in ρ is a* transition state *of θ if $\theta \not\subseteq \sigma_{j-1}$ and $\theta \subseteq \sigma_j$.*

The state σ_j is a transition state of θ if the outcome is satisfied in σ_j but not in the immediately previous state σ_{j-1}. It is easy to see that if 4 is satisfied for some σ_j in ρ, then by the successor state equation 2.4 it must be the case that one or more elementary events in ϵ_{j-1} has caused at least one of θ's literals to hold by σ_j. Note that there may be multiple transition states for an outcome θ in a given path ρ.

In our running example, σ_4 is the only transition state of θ_E because it is only state of ρ_E in which the literals A, B, C, D, E and F hold simultaneously. It is easy to verify that $\theta_E \not\subseteq \sigma_3$ and $\theta_E \subseteq \sigma_4$ in ρ_E using Table 3.1.

Given a transition state σ_j and a literal l in outcome θ, we can identify the most recent compound event to σ_j in ρ to result in l. In other words, we want to find a *causing compound event* ϵ_i that resulted in the most recent transition state of the singleton $\{l\}$ with respect to θ's transition state σ_j. We first provide a preliminary definition for a *possibly causing compound* event.

Definition 5. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a transition state σ_j of θ in ρ, and a literal $l \in \theta$, ϵ_i is a* possibly causing compound event *of l for σ_j if the state σ_{i+1} in ρ is a transition state of $\{l\}$ in ρ and $i < j$.*

Now we may define causing compound events.

Definition 6. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a transition state σ_j of θ in ρ, a literal $l \in \theta$, and a possibly causing compound event ϵ_i of l for σ_j, ϵ_i is a* causing compound event *of l for σ_j if there is no other possibly causing compound event $\epsilon_{i'}$ for l in σ_j such that $i < i'$.*

It is easy to see that if there is no causing compound event of $l \in \theta$ for a transition state σ_j, then l must have held in the initial state of ρ and was never changed by a subsequent event prior to σ_j.

It is straightforward to verify using Table 3.1 that ϵ_1 is a causing compound event of A, ϵ_2 is a causing compound event of B and C, and ϵ_3 is a causing compound event of D, E, and F. In all cases, Definition 6 is satisfied because σ_2 is a transition state of A, σ_3 is a transition state of B and C, and σ_4 is a transition state of D, E, and F and there are no other transition states for any literal in $l \in \theta_E$. Therefore, there cannot be a more recent causing compound event of for any such l holding in transition state σ_4 of θ_E.

3.2.2 Direct Cause

Once we know that ϵ_i is a causing compound event for l in σ_i, we can "look inside" of ϵ_i to identify direct and/or indirect causes of l.

Definition 7. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a transition state σ_j of θ in ρ, a literal $l \in \theta$, and a causing compound event ϵ_i of l, the elementary event $e \in \epsilon_i$ is a* direct cause *of l for σ_j if $l \in E(e, \sigma_i)$.*

If l is in the set of direct effects of e occurring in state σ_j, then e's occurrence was sufficient to directly cause l. Note that direct cause is defined in such a way that multiple events can be direct causes simultaneously as long as l is in the corresponding sets of direct effects. For example, we already know that ϵ_1 is a causing compound event of A holding in σ_4 in the example. Definition 7 tells us that $e_1 \in \epsilon_1$ is a direct cause of A for σ_j. This is because A is in the set $E(e_1, \sigma_1)$ due to the dynamic law (3.1) of AD_E. It can be similarly verified that $e_2 \in \epsilon_1$ is also a direct cause of A because the literal is in $E(e_2, \sigma_1)$. We can also verify that $e_3 \in \epsilon_2$ is an direct cause of C, and finally $e_4 \in \epsilon_3$ and $e_5 \in \epsilon_3$ are direct causes of E and F, respectively.

3.2.3 Simple Notion of Indirect Cause

In [48], an *indirect cause* of a literal l is a *subset* of elementary events in $\epsilon \subseteq \epsilon_i$ that have indirectly caused the literal [1].

Definition 8. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a transition state σ_j of θ in ρ, a literal $l \in \sigma_j$, and a causing compound event ϵ_i of l, the compound event $\epsilon \subseteq \epsilon_i$ is an* indirect cause *of l for σ_j if it is a smallest subset of ϵ_i such that the following conditions are satisfied:*

1. $l \notin E(\epsilon, \sigma_i)$
2. *There exists a transition $t = \langle \sigma_i, \epsilon, \sigma'_{i+1} \rangle$ in $\tau(AD)$ such that σ'_{i+1} is a transition state of $\{l\}$ in t*

Note that condition 1 requires that l is not a direct effect of ϵ. Condition 2 checks that if ϵ were to hypothetically occur by itself in state σ_i, then l would hold in the resulting state. Finally, we require that ϵ is a *smallest* subset of ϵ_i because we want to rule out any subsets including extraneous elementary events. For example, if ϵ contains three events and only two are capable of causing l, then there would certainly be a transition $t = \langle \sigma_i, \epsilon, \sigma'_{i+1} \rangle$ as required by condition 2, but we want ϵ to contain only event(s) that have indirectly caused l.

In our example, $\{e_3\} \subseteq \epsilon_2$ is an indirect cause of B for σ_4 because ϵ_2 is a causing compound event of B and $\langle \sigma_2$, B is not in the set $E(\{e_3\}, \sigma_2)$, and finally $\langle \{e_3\}, \sigma' \rangle$ is a valid transition where σ' is a transition state of $\{B\}$ as per laws (3.3) and (3.7). It can be similarly verified that $\{e_4, e_5\}$ is an indirect cause of D for σ_4.

Problems with the Hypothetical Reasoning Approach

Although we have found that Definition 8 can be used to solve some traditionally challenging examples from the literature [48], the following counterexample points out a fundamental problem with the approach to verifying indirect cause using Definition 8. Consider the following action

[1] In $\mathcal{AL}$, it is possible that a set of literals must hold simultaneously in order for l to be caused (e.g. see Law (3.8) in AD_E).

description:

$$e_1 \textbf{ causes } d_1 \tag{3.9}$$

$$e_2 \textbf{ causes } d_2 \tag{3.10}$$

$$l \textbf{ if } d_1 \tag{3.11}$$

$$l \textbf{ if } d_2, \neg d_1 \tag{3.12}$$

Consider now a path ρ with a single transition $\langle \sigma, \epsilon, \sigma' \rangle$ in which $\neg d_1$, $\neg d_2$, and $\neg l$ hold in the initial state, and $\epsilon = \{e_1, e_2\}$. The occurrence of ϵ causes d_1, d_2, and l to hold in σ'. It is straightforward to verify that that e_1 is a direct cause of d_1 in σ' as per law (3.9), and similarly that e_2 is a direct cause of d_2 via law (3.10). Using Definition 8, we find that $\{e_1\}$ and $\{e_2\}$ are overdetermining indirect causes of l for σ_1. In the case of $\{e_1\}$, this is intuitive because d_1 was caused to hold by e_1 and l is a ramification of this event by law (3.9). However, identifying $\{e_2\}$ as an indirect cause is *not* in line with intuition because d_1 holds in σ' in the actual scenario, so $\{e_2\}$ does not appear to have had influence on l. Indeed, if e_2 were to occur by itself in σ, it would indirectly cause l to hold because the preconditions of law (3.12) would be satisfied, but e_1's membership in ϵ caused d_1 to hold in σ' and so (3.12) does not actually "apply" in σ'. This leads us to the conclusion that Definition 8 incorrectly identifies events as indirect causes when the preemption of the events effects occurs over a single transition. We refer to such a case as *single-transition preemption*.

An additional problem identified by our analysis of Definition 8 is the condition that ϵ be the smallest subset of ϵ_i satisfying conditions 1 and 2. Consider the following action description:

$$e_1 \textbf{ causes } d_1 \tag{3.13}$$

$$e_1 \textbf{ causes } d_2 \tag{3.14}$$

$$e_2 \textbf{ causes } d_3 \tag{3.15}$$

$$l \textbf{ if } d_1 \tag{3.16}$$

$$l \textbf{ if } d_2, d_3 \tag{3.17}$$

Consider now a path ρ with a single transition $\langle \sigma, \epsilon, \sigma' \rangle$ in which $\neg d_1$ and $\neg d_2$ hold in the initial state, and $\epsilon = \{e_1, e_2\}$. The occurrence of ϵ causes d_1 and d_2 to hold in σ'. It is straightforward to verify that that e_1 is a direct cause of d_1 in σ' as per law (3.9), and similarly that e_2 is a direct cause of d_2 via law (3.10). In this case, Definition 8 would identify $\{e_1\}$ as an indirect cause, but would miss the intuitive answer that $\{e_1, e_2\}$ is an overdetermining indirect cause with $\{e_1\}$ due to rules (3.14), (3.15), and (3.17). We can conclude that the smallest subset condition is too restrictive and causes the definition to miss intuitive cases of indirect causation. We refer to such a case as *larger subset overdetermination*.

In addition to producing non-intuitive results in these cases, these examples bring to light the fact that the hypothetical reasoning step is not philosophically in line with reasoning about what *actually* happened. Therefore, we have developed a new definition of indirect cause that links the direct effects of events to the indirect causation of a literal l via the state constraints of the action description.

3.2.4 Improved Definition of Indirect Cause

The improved definition of indirect cause requires some preliminary notions. First, a *link* γ is a set of state constraints. The set of all consequences in γ is denoted by $C(\gamma)$ and the set of all preconditions is $P(\gamma)$. The set of all consequences of a set of links is denoted by $C(\{\gamma_1, \ldots, \gamma_n\}) = C(\gamma_1) \cup \ldots \cup C(\gamma_k)$. Similarly, the set of all preconditions of a set of links is $P(\{\gamma_1, \ldots, \gamma_n\}) = P(\gamma_1) \cup \ldots \cup P(\gamma_k)$.

Given a transition $t = \langle \sigma_i, \epsilon_i, \sigma_{i'} \rangle$, the set of *preserved preconditions* of a state constraint s for σ'_i is given by $I(s, \sigma_i) = (\sigma_{i'} \cap \sigma_i) \cap prec(s)$. In other words, these are the preconditions of state constraint s that were preserved by inertia in the transition, rather than being caused as a direct effect or ramification of ϵ_i. The set of s's *directly caused preconditions* by $\epsilon \subseteq \epsilon_i$ for σ_i is the set $M(s, \epsilon, \sigma'_i) = E(\epsilon, \sigma_i) \cap prec(s) \setminus \sigma_i$. This condition reflects our position that a literal l that was preserved by inertia from σ_i to σ'_i should not be considered to have been caused by any subset of ϵ_i even if l is in the set of direct effects $E(\epsilon_i, \sigma_i)$.

A *static chain* is a sequence of links that connects the direct effects of one or more elementary events in a causing compound event ϵ_i of l to the indirect causation of l in transition state σ'_i of $\{l\}$

by way of the state constraints of AD.

Definition 9. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a compound event $\epsilon \subseteq \epsilon_i$ in ρ, sequential states σ_i and $\sigma_{i'}$ in ρ, and a literal $l \in \sigma_{i'}$, a* static chain *$\chi(\epsilon, l, \sigma_{i'}) = \langle \gamma_1, \ldots, \gamma_n \rangle$ is a sequence of non-empty links where l is only in the consequences of the final link γ_n, and such that the links of $\chi(\epsilon, l, \sigma_{i'})$ satisfy the following conditions:*

1. *γ_1 is the set of all state constraints s such that $prec(s) \subseteq \sigma_{i'}$, and there exists a non-empty set $M \subseteq M(s, \epsilon, \sigma_{i'})$ such that $prec(s) = M \cup I(s, \sigma_{i'})$.*

2. *if $l \notin C(\gamma_{g-1})$ for every $1 < g \leq n$, γ_g is the set of state constraints s such that:*

   (a) $prec(s) \subseteq \sigma_{i'}$, and

   (b) $prec(s) = C(\gamma_{g-1}) \cup C(\{\gamma_1, \ldots, \gamma_{g-2}\}) \cup M' \cup I(s, \sigma_{i'})$,

   where $C(\gamma_{g-1})$ is a non-empty set and $M' \subseteq M(s, \epsilon, \sigma_{i'})$.

Condition 1 states that the first link of a causal chain contains only those state constraints whose preconditions became satisfied as a result of the direct effects of ϵ. Condition 2 says that any state constraint in link γ_g must be "connected" to the immediately previous link γ_{g-1} by at least one member of γ_{g-1}'s consequences. Moreover, l may not have already been caused as a consequence of γ_{g-1}. This requirement along with the requirement that l is only a consequence of the final link γ_n allows us to define when a static chain "terminates".

We can identify a number of interesting properties for static chains.

Proposition 2. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a compound event ϵ_i, a literal $l \in \theta$, and sequential states σ_i and $\sigma_{i'}$, there is exactly one static chain $\chi(\epsilon, l, \sigma_{i'})$ for a compound event $\epsilon \in \epsilon_i$.*

This can be easily verified by considering the fact there are no non-deterministic effects of actions in $\mathcal{AL}$, and therefore the direct effects of ϵ can be linked to l in exactly one way. Therefore, when we identify a static chain $\chi(\epsilon, l, \sigma_{i'})$, we can be assured that there are no additional chains $\chi'(\epsilon, l, \sigma_{i'})$ that would explain the indirect causation of l in $\sigma_{i'}$. Another interesting property is that a static chain contains at least one link. Formally,

Proposition 3. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a compound event ϵ_i in ρ, a literal $l \in \theta$, and a static chain $\chi(\epsilon, l, \sigma_i') = \langle \gamma_1, \ldots, \gamma_n \rangle$ for some $\epsilon \in \epsilon_i$, $n \geq 1$.*

It is clear from the definition of a static chain that $\chi(\epsilon, l, \sigma_i')$ has at least one link. Finally, it can be verified that the length of a static chain is finite when the number of state constraints in the action description is finite. Formally,

Proposition 4. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a compound event ϵ_i in ρ, a literal $l \in \theta$, and a static chain $\chi(\epsilon, l, \sigma_i') = \langle \gamma_1, \ldots, \gamma_n \rangle$ for some $\epsilon \in \epsilon_i$, the maximum length of $\chi(\epsilon, l, \sigma_i')$ is equivalent to the number of state constraints in AD.*

The following proposition will be useful in our proof of correctness of the implementation in Chapter 5.

Proposition 5. *Given a static chain $\chi(\epsilon, l, \sigma_i) = \langle \gamma_1, \ldots, \gamma_n \rangle$ and a state constraint s:*

- *if $s \in \gamma_1$, then $|prec(s)| = |M| + |(s, \sigma_{i'})|$*
- *if $s \in \gamma_g$, $g < 1 \leq n$, and $C(\{\gamma_1, \ldots, \gamma_{g-2}\}) \cap C(\gamma_{g-1}) = \emptyset$, then $|prec(s)| = |C(\{\gamma_1, \ldots, \gamma_{g-2}\})| + |\gamma_{g-1}| + |M'| + |I(s, \sigma_{i'})|$*

Now we may leverage the definition of static chain to identify when a proper subset $\epsilon \subseteq \epsilon_i$ is an indirect cause of l for state σ_i'.

Definition 10. *Given a problem $\psi = \langle \theta, \rho, AD \rangle$, a literal $l \in \theta$, a transition state σ_j of θ in ρ, sequential states σ_i and $\sigma_{i'}$ in ρ, and a causing compound event ϵ_i of l for σ_j, $\epsilon \subseteq \epsilon_i$ is an* indirect cause *of l in σ_j if the following conditions hold:*

1. *There exists a static chain $\chi(\epsilon, l, \sigma_{i'}) = \langle \gamma_1, \ldots, \gamma_n \rangle$.*
2. *There exists no event in $e \subseteq \epsilon$ such that $E(e, \sigma_i) \cap P(\{\gamma_1, \ldots, \gamma_n\}) = \emptyset$ for $\chi(\epsilon, l, \sigma_{i'})$.*

Condition 1 requires that there is a static chain linking the direct effects of ϵ to l in σ_i by way of the state constraints of AD. Condition 2 states that there is no elementary event in ϵ' whose direct effects do not contribute to the satisfaction of any state constraint in any γ_i of the static chain under

consideration. This allows us to exclude extraneous events from our definition of indirect cause. Note that it is impossible for an empty set to be an indirect cause of l in σ_i due to Condition 1 of the definition of a static chain stating that M is a non-empty set.

Now, we will return to the running example and use Definitions 9 and 10 to once again identify $\{e_3\}$ as an indirect cause of B and $\{e_4, e_5\}$ as an indirect cause of D. First, there exists a static chain $\chi(\{e_3\}, B, \sigma_3) = \langle\gamma_1\rangle$ linking the direct effects of $\{e_3\}$ to B for state σ_3. Let s represent law (3.7). In this case condition 1 of Definition 9 is satisfied for s because $prec(s) \subseteq \sigma_3$ and $prec(s) = M(s, \{e_3\}, \sigma_3) \cup I(s, \sigma_3)$ where $M(s, \{e_3\}, \sigma_3) = \{C\}$ by law (3.3) and $I(s, \sigma_3) = \emptyset$. Therefore, $s \in \gamma_1$. Because $s \in \gamma_1$ and $consq(s) = B$, the literal B is clearly in $C(\gamma_1)$. By condition 2, there can be no link beyond γ_1 and Definition 9 is satisfied. Finally, because there are no extraneous events in $\{e_3\}$, this event is an indirect cause of B for σ_4 by condition 2 of Definition 10. It can be similarly verified that there is a chain $\chi(\{e_4, e_5\}, B, \sigma_3) = \langle\gamma_1\rangle$ where γ_1 contains state constraint (3.8), and moreover that $\{e_4, e_5\}$ is an indirect cause of D for σ_4 because there are no extraneous events in $\{e_4, e_5\}$ as per laws (3.4), (3.5), and (3.8).

Single-Transition Preemption Revisited

Consider again the first counterexample to Definition 8 from Section 3.2.3. In this case, we previously incorrectly identified e_2 as an indirect cause of l. Let s represent law (3.11). Using Definition 9 for static chains, it can be verified that there is a static chain $\chi(\{e_1\}, l, \sigma') = \langle\gamma_1\rangle$ such that $s \in \gamma_1$. Here, s satisfies condition 1 of Definition 9 because its only precondition d_1 holds in σ' and d_1 is also a direct effect of $\{e_1\}$. The definition of static chain is satisfied because l is a member of the set of consequences of γ_1. Because there is clearly no extraneous event in $\{e_1\}$, $\{e_1\}$ is an indirect cause of l for σ'. There is no chain $\chi(\{e_2\}, l, \sigma')$ because the direct effect d_2 of e_2 in σ_1 is not a precondition of any law whose full set of preconditions hold in σ_2. Therefore $\{e_2\}$ is not an indirect cause of l for σ'.

Larger Subset Overdetermination Revisited

Consider now the second counterexample to Definition 8 from Section 3.2.3. Here, we previously missed an intuitive indirect cause due to the smallest subset requirement of the original definition of indirect cause. Using Definition 9, it is straightforward to verify that there are static chains linking both $\{e_1\}$ and $\{e_1, e_2\}$ to d_1 in σ'. Let s represent the state constraint (3.16). Static chain $\chi(\{e_1\}, d_1, \sigma')$ contains a link $\gamma_1 = \{s\}$ because s's precondition d_1 holds in σ' and is a direct effect of e_1 occurring in σ_1. Now let s' represent law 3.17. The static chain $\chi(\{e_1, e_2\}, d_1, \sigma')$ contains a link $\gamma_1 = \{s\}$ because the set of preconditions $prec(s') \in \sigma'$ and both literals were caused directly by $\{e_1, e_2\}$ (i.e., $d_2 \in E(e_1, \sigma)$ and $d_3 \in E(e_2, \sigma)$). It is easy to see that there are no extraneous events in either chain, and so $\{e_1\}$ and $\{e_1, e_2\}$ are overdermining indirect causes of l for σ'.

3.3 Discussion

There are several technical and conceptual advantages of the improved definition of indirect cause over the original. The most attractive advantage is the information contained in static chains. Not only is a static chain a useful tool for identifying *when* a literal has been caused as a ramification of a compound event, but the chain itself also tells the story of exactly *how* the literal is linked to the direct effects of the event. This puts us at a clear explanatory advantage over approaches that define actual causation in terms of dependence because even if they reach the same conclusions as us about the cause of some outcome of interest, they are unable to explain how the outcome was influenced – only that it *was* influenced.

Another advantage of Definition 10 is that we have been able to lift the restriction enforced by condition 1 of Definition 8 that an event cannot be both a direct and indirect cause. It is easy to imagine a situation in which this is possible, for example:

$$e_1 \textbf{ causes } d_1 \tag{3.18}$$

$$e_1 \textbf{ causes } d_2 \tag{3.19}$$

$$d_1 \textbf{ if } d_2 \tag{3.20}$$

Consider that $\neg d_1$ and $\neg d_2$ both hold in σ_i and that $\epsilon_i = \{e_1\}$. In this case, d_1 is a direct effect of e_1 because $d_1 \in E(e_1, \sigma_i)$ as per law (3.18). However, the subset $\{e_1\}$ is also an indirect cause of d_1. Due to law (3.19), there is a static chain $\chi(\{e_1\}, d_1, \sigma_\prime) = \langle \gamma_1 \rangle$ such that the state constraint (3.20) is the only member of γ_1, and there are no extraneous events in $\chi(\{e_1\}, d_1, \sigma_\prime)$. Because there are no extraneous events, $\{e_1\}$ is an indirect cause of d_1 for σ'. The original definition can only be used to identify e_1 as a direct cause of d_1, and will miss its indirect causation of d_1.

Definition 10 also has an important conceptual advantage over the original, which is that it requires reasoning strictly over what has actually happened, rather then checking to see if there is a possible world in which ϵ can cause a literal to hold in some state $\sigma_{i'}$. By only considering the state constraints whose preconditions are satisfied in a state $\sigma_{i'}$, we automatically rule out any events whose effects were blocked in the transition, as earlier demonstrated.

There are a number of important insights about this approach to be gained from the presentation. The first is that the representation of scenarios as the evolution of state using brings with it natural solutions to situations that are traditionally challenging to counterfactual reasoning approaches, as we have shown. Another is that scenarios as paths are intuitive and iconic - that is to say, paths more closely reflect how we mentally represent scenarios as events causing changes to to the state of the world as opposed to a set or sequence of events. Finally, having the ability to reason over the semantics of $\mathcal{AL}$, the elements of a path, and the causal knowledge contained in the action description AD frees us from reasoning about *possible* worlds by giving us the information we need to reason about the *actual* world. We address these advantages in greater detail in Chapter 7.

In the next chapter, we test the framework on a number of examples from the literature, as well as a novel example inspired by the self-driving car example introduced in Chapter 1.

Chapter 4: Examples

In this chapter, we test the framework on a number of examples from the literature, as well as a novel example concerning a crash scenario in the context of a self-driving car. By testing the framework on the literature examples, we intend to demonstrate that our framework is able to match or exceed the intuition gained using a counterfactual dependence as a condition for cause. In proposing the self-driving car example, we aim to demonstrate, albeit at a high level, that it is possible to reason about independently operating components in a theoretical distributed system.

4.1 Scenario Paths

We represent the elements of each example using an *example tuple* $\Psi = \langle \theta, v, AD \rangle$, where θ is an outcome of interest, v is a sequence of compound events representing the events of the example, and AD is an action description of the example's domain. In order to ensure that our approach is starting with the same ingredients as other approaches to reasoning about actual cause (events representing a scenario instead of a path), we define *scenario paths* to map the sequence of compound events v to one or more paths of the transition diagram $\tau(AD)$. Scenario paths represent a unique unfolding of a scenario's events with respect to a given domain and provide a convenient representation of how the domain changes over time in response to the events of the scenario. We reason over these paths to explain actual causation.

Definition 11. *Given an outcome θ, a sequence of events $v = \langle \epsilon_1, \ldots, \epsilon_k \rangle$, and an action description AD, a* scenario path *is a path $\rho = \langle \sigma_1, \epsilon_1, \sigma_2, \epsilon_2, \ldots, \epsilon_k, \sigma_{k+1} \rangle$ in $\tau(AD)$ satisfying the following conditions:*

1. $\theta \neq \sigma_1$
2. $\exists i, 1 < i \leq k+1, \theta \subseteq \sigma_i$

Condition 1 requires that the set of fluent literals θ is not satisfied in the initial state of ρ, ensuring that the outcome has not already been caused prior to the known events of the story. Condition 2

requires that θ is satisfied in at least one state after the initial state in ρ. Conditions 1 and 2 together ensure that at least one event is responsible for causing θ to hold in ρ. The successor state equation (2.4) tells us some event in the scenario path must have directly or indirectly caused θ to be satisfied at some point after the initial state. The set of all scenario paths for an example tuple Ψ is given by $S(\Psi) = \{\rho_1, \rho_2, \ldots, \rho_m\}$. A problem $\psi = \langle \theta, \rho, AD \rangle$ where $\rho \in S(\Psi)$ allows us to inquire about causal information for a specific path rather than for a sequence of events as in Ψ.

4.1.1 Rock Throwing Problem

First, we will explore two versions of the rock-throwing problem [49], demonstrating that our approach can identify causation in the original example of preemption, and a reformulation of the problem exhibiting overdetermination [50].

Preemption

The original scenario posed in [49] is as follows:

> *Throwing a rock at a bottle will cause it to break. Suzy throws the rock at a bottle, and Billy throws a second rock at the bottle. Suzy's rock hits first and the bottle is broken. Who is to blame for the bottle's breaking?*

Intuition tells us that Suzy is responsible for breaking the bottle. We can build upon this example to say that Tommy handed Suzy the rock at the start of the scenario. We extend the scenario to capture these dynamics as follows:

> *Handing a rock to Suzy causes it to be in her possession. Throwing a rock at a bottle will cause it to break. Tommy hands a rock to Suzy, she throws the rock at a bottle, and Billy throws a second rock at the bottle. Suzy's rock hits first and the bottle is broken. Who is to blame for the bottle's breaking?*

Mapping this example to a practical setting, say a vandalism trial, we would certainly want to blame Suzy for succeeding in breaking the bottle, and possibly Tommy as well for his action of

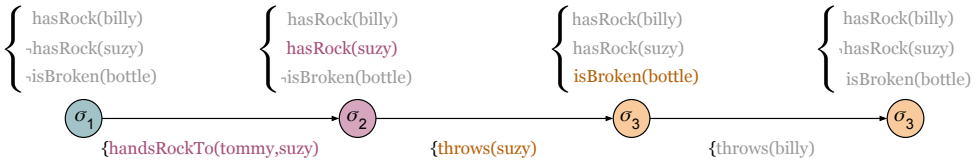

Figure 4.1: Path $\rho_B \in S(\Psi_B)$ is a representation of the bottle breaking scenario.

handing the rock to Suzy [1].

The elements of the problem are given by the example tuple $\Psi_Y = \langle \theta_B, v_B, AD_B \rangle$. In this example the outcome of interest is given by $\theta_B = \{isBroken(bottle)\}$. The action description AD_B characterizes the events of the bottle breaking domain:

$$\begin{cases} handsRockTo(tommy, suzy) \textbf{ causes } hasRock(suzy) & (4.1) \\ throws(suzy) \textbf{ causes } isBroken(bottle) & (4.2) \\ \quad \textbf{if } \neg isBroken(bottle) & \\ throws(billy) \textbf{ causes } isBroken(bottle) & (4.3) \\ \quad \textbf{if } \neg isBroken(bottle) & \\ throws(suzy) \textbf{ causes } \neg hasRock(suzy) & (4.4) \\ throws(suzy) \textbf{ impossible\_if } \neg hasRock(suzy) & (4.5) \end{cases}$$

Law (4.1) tells us that if Tommy hands a rock to Suzy, then Suzy has the rock. Laws (4.2) and (4.3) represent the knowledge that if someone throws a rock at a bottle, it will break. Law (4.4) tells us that if Suzy throws the rock, then she no longer is in possession of the rock. Finally, Law (4.5) states that it is impossible for Suzy to throw a rock if it is not in her possession. We assume that there exists a corresponding law for Billy for each of (4.1), (4.4), and (4.5), however we omit them for clarity of the presentation. Consider a path $\rho_B \in S(\Psi_B)$ corresponding to the scenario's events,

[1] The matter of Billy's intention to commit the crime is a matter of judging levels of blame (see e.g. [51]), which is outside of the scope of this work.

represented in Figure 4.1. In the initial state of ρ_B, Suzy does not have the rock and the bottle is not broken. After the occurrence of $\epsilon_1 = \{handsRock(tommy, suzy)\}$, $\epsilon_2 = \{throws(suzy)\}$, and $\epsilon_3 = \{throws(billy)\}$, the bottle is broken in state σ_4. It is straightforward to verify for problem $\psi_B = \langle \theta_B, \rho_B, AD_B \rangle$ that σ_3 is the only transition state of θ_B in ρ_B and that ϵ_2 is the causing compound event $isBroken(bottle) \in \theta_B$ holding in σ_3. The elementary event $throws(suzy)$ is a direct cause of $isBroken(bottle)$ because it is in the set $E(throws(suzy), \sigma_1)$ as per rule (4.2).

There is more causal information to uncover in this problem. We already know that Suzy throwing the rock at the bottle caused it to break, but we want to know if any events supported Suzy's ability to cause the outcome. Rule (4.1) in AD_B tells us that Suzy cannot throw the rock if it is not in her possession. In this case, we can formulate a new problem $\psi'_Y = \langle hasRock(suzy), \rho_B, AD_B \rangle$ and use the framework to determine that $handsRock(tommy, suzy)$ directly caused $hasRock(suzy)$. Table 4.1 summarizes the causal explanations for each of the outcomes we considered. Each row of the first column gives the outcomes of interest for the problems ψ_Y and ψ'_Y. The second column identifies the transition state for each outcome in ρ_B. The third column gives the direct causes for both problems. It is easy to see that there are no indirect causes for this example.

We have shown that modeling the scenario as a sequence of events occurring over time enables the framework to correctly identify the intuitive cause of the bottle breaking in a classic example of preemption from the literature. We have also shown that it is straightforward to reformulate the problem with a new outcome in order to learn more about the causal mechanism at will.

Overdetermination

The scenario can be modified so that Suzy and Billy throw their rocks at the same time, both hitting the bottle simultaneously, upon which it shatters as in [50]. Either rock by itself would have sufficed to shatter the bottle, so we want to identify both throws as direct causes. Omitting the extension in which Tommy hands the rock to either Suzy or Billy, we substitute v_B in Ψ_B with the event sequence $v'_B = \langle \epsilon_1 \rangle$ where $\epsilon_1 = \{throws(suzy), throws(billy)\}$. This yields a new scenario path ρ'_B and the resulting problem for the framework is $\psi'_B = \langle \theta_B, \rho'_B, AD_B \rangle$. It is straightforward to verify that both $throws(suzy)$ and $throws(billy)$ are direct causes for $isBroken(bottle)$ in ρ'_B

Table 4.1: Overview of explanations of $\{hasRock(suzy)\}$ and $\{isBroken(bottle)\}$ in transition states σ_2 and σ_3, respectively.

Outcome of Interest	*State*	*Direct Causes*
$\{isBroken(bottle)\}$	σ_4	$throws(suzy) \in \epsilon_2$
$\{hasRock(suzy)\}$	σ_2	$handsRockTo(tommy, suzy) \in \epsilon_1,$

because $isBroken(bottle)$ is a member of both $E(throws(suzy), \sigma_1)$ and $E(throws(billy)), \sigma_1)$. As before, there are no indirect causes for outcome in the new problem. We have demonstrated that the framework can identify direct causes that match our intuition in this example of overdetermination.

4.2 Yale Shooting Problem

4.2.1 Direct

Here we use the framework defined above to solve a variant of the well-known Yale shooting problem (YSP) from [52]. The scenario is as follows:

> *Shooting a turkey with a loaded gun will kill it. Suzy shoots the turkey. What is the cause of the turkey's death?*

The YSP example tuple is formalized by $\Psi_Y = \langle \theta_Y, v_Y, AD_Y \rangle$. The outcome of interest is $\theta_Y = \{\neg isAlive(turkey)\}$. The sequence of events is $v_Y = \{\epsilon_1, \epsilon_2\}$, where $\epsilon_1 = \{loads(suzy, gun)\}$ and $\epsilon_2 = \{shoots(suzy, turkey)\}$. The action description AD_Y characterizes the events of the YSP domain:

$$\begin{cases} shoots(X, turkey) \textbf{ causes } \neg isAlive(turkey) \textbf{ if } isAlive(turkey) & (4.6) \\ shoots(X, turkey) \textbf{ impossible\_if } \neg isLoaded(gun) & (4.7) \\ \quad loads(X, gun) \textbf{ causes } isLoaded(gun) \textbf{ if } \neg isLoaded(gun) & (4.8) \end{cases}$$

Laws (4.6) and (4.8) are straightforward dynamic laws describing the effects of the events in the YSP domain. Law (4.7) states that the turkey cannot be shot if the gun is not loaded. Consider the path ρ_Y, represented in Figure 4.2. In the initial state of ρ_Y, the turkey is alive, and the turkey

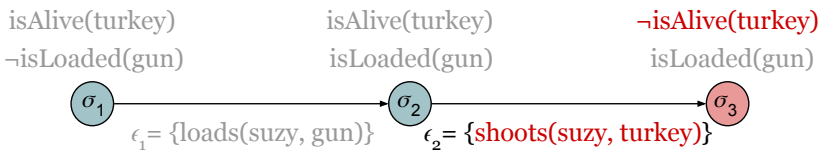

Figure 4.2: Path $\rho_Y \in S(\Psi_Y)$ is a representation of the Yale shooting scenario.

is dead in the final state of the path after the occurrence of $\epsilon_1 = \{loads(suzy, gun)\}$ and $\epsilon_2 = \{shoots(suzy, turkey)\}$.

It is straightforward to verify for the problem $\psi_Y = \langle \theta_Y, \rho_Y, AD_Y \rangle$ that σ_3 is the only transition state of θ_Y and that ϵ_2 is the causing compound event of $\neg isAlive(turkey)$. The elementary event $shoots(suzy, turkey)$ in ϵ_2 is a direct cause of $\neg isAlive(turkey)$ as per rule (4.6). If we want to know why the gun was loaded so that Suzy could kill the turkey, we can use rule (4.7) to formulate the problem $\psi'_Y = \langle \{isLoaded(gun)\}, \rho_Y, AD_Y \rangle$ to determine that $loads(suzy, gun)$ directly caused the gun to be loaded.

4.2.2 Indirect

Here we use the framework definitions to solve a novel adaptation of the well-known Yale shooting problem (YSP). The scenario is as follows:

> *Shouting near a turkey will startle the turkey, and the turkey will die if it is startled. Suzy walks to the turkey and shouts. What is the cause of the turkey's death?*

In this case, we want to identify Suzy shouting near the the turkey is an indirect cause of its death. The elements of the indirect YSP problem are given by $\Psi_Y = \langle \theta_Y, v_Y, AD_Y \rangle$. The outcome of interest is $\theta_Y = \{\neg isAlive(turkey)\}$. The sequence of events is given by $v_Y = \{\epsilon_1, \epsilon_2\}$ where $\epsilon_1 = \{walksTo(suzy, turkey)\}$ and $\epsilon_2 = \{shouts(suzy)\}$. The action description AD_Y characterizes the events of the YSP domain:

$$\begin{cases} shouts(X) \textbf{ causes } isStartled(turkey) \textbf{ if } \neg isStartled(turkey) & (4.9) \\ \neg isAlive(turkey) \textbf{ if } isStartled(turkey) & (4.10) \end{cases}$$

Laws (4.9) and (4.10) represent the domain knowledge for this example in a straightforward way. Consider the scenario path $\rho_Y \in S(\Psi_Y)$, represented in Figure 4.3. In the initial state of ρ_Y, the turkey is alive, and the turkey is dead in the final state of the path after the occurrence of $\epsilon_1 = \{walksTo(suzy, turkey)\}$ and $\epsilon_2 = \{shouts(suzy)\}$.

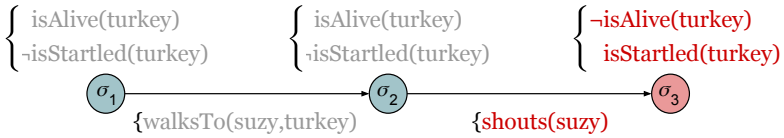

Figure 4.3: Path $\rho_Y \in S(\Psi_Y)$ is a representation of the Yale shooting scenario.

It is easy to see that for the new problem $\psi_Y = \langle \theta_Y, \rho_Y, AD \rangle$, σ_4 is the only transition state of θ_Y and that ϵ_2 is the causing compound event of $\neg isAlive(turkey)$. Let s represent law (4.10). It is straightforward to verify that there is a static chain $\chi(\{shouts(suzy)\}, \neg isAlive(turkey), \sigma_3) = \langle \gamma_1 \rangle$ where $s \in \gamma_1$. Because $\neg isDead(turkey) \in C(\gamma_1)$, Definition 10 is satisfied. Finally, because there are no extraneous events in $\chi(\{shouts(suzy)\}, \neg isAlive(turkey), \sigma_3)$, $\{shouts(suzy)\}$ is an indirect cause of $\neg isAlive(turkey)$ for state σ_3. We have used the framework to correctly identify only Suzy's shouting as an indirect cause of its death. Moreover, we correctly identified zero direct causes.

4.2.3 Novel Extension

As we have already discussed, the intuitive cause of the turkey's death is that Suzy shot the turkey. However, if we know for certain that the gun was not loaded at the start of the story, then we argue that it may also important to recognize that Suzy loading the gun played a key role in bringing about the outcome. Mapping this example to a practical setting, say a murder trial, Suzy loading the gun could contribute to evidence of Suzy's intention to shoot the turkey. Similarly to how we built upon the Rock Throwing example in Chapter 4, if we extend this example to say that Tommy handed Suzy the gun at the start of the scenario, then again we want to identify Tommy's action as contributing to the turkey's death. The extended example as follows:

Tommy hands a gun to Suzy, Suzy loads the gun, and then shoots the turkey. What is the cause

of the turkey's death?

The elements of the extended Yale shooting problem are given by the example tuple $\Psi_Y = \langle \theta_Y, v_Y, AD_Y \rangle$. The outcome of interest is represented by $\theta_Y = \{\neg isAlive(turkey)\}$. The sequence of events corresponding to the Yale shooting scenario is given by $v_Y = \{\epsilon_1, \epsilon_2, \epsilon_3\}$ where $\epsilon_1 = \{handsGun(tommy, suzy)\}$, $\epsilon_2 = \{loads(suzy, gun)\}$, and $\epsilon_3 = \{shoots(suzy, turkey)\}$. The action description AD_Y characterizes the events of the bottle breaking domain:

$$
\begin{cases}
shoots(suzy, turkey) \textbf{ causes } \neg isAlive(turkey) & (4.11) \\
\qquad \textbf{if } isAlive(turkey) & \\
shoots(suzy, turkey) \textbf{ impossible\_if } \neg isLoaded(gun) & (4.12) \\
loads(suzy, gun) \textbf{ causes } isLoaded(gun) \textbf{ if } \neg isLoaded(gun) & (4.13) \\
loads(suzy, gun) \textbf{ impossible\_if } isLoaded(gun) & (4.14) \\
loads(suzy, gun) \textbf{ impossible\_if } \neg hasGun(suzy) & (4.15) \\
handsGunTo(tommy, suzy) \textbf{ causes } hasGun(suzy) & (4.16)
\end{cases}
$$

Laws (4.11), (4.13), and (4.16) are straightforward dynamic laws describing the effects of the events in the YSP domain. Law (4.7) states that the turkey cannot be shot if the gun is not loaded. Finally, law (4.12) says that Suzy cannot load an already loaded gun, and law (4.15) states that X cannot load the gun if they do not have it in their possession, modeling some simple knowledge about the gun.

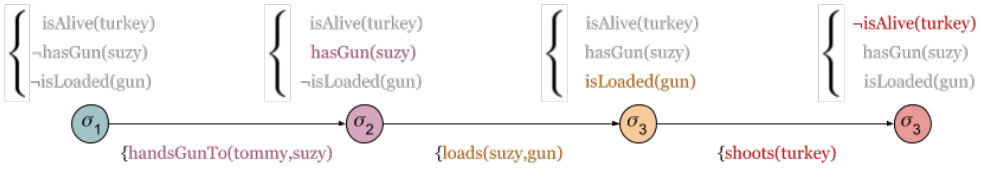

Figure 4.4: Path $\rho_Y \in S(\Psi_Y)$ is a representation of the Yale shooting scenario.

Consider the scenario path $\rho_Y \in S(\Psi_Y)$, represented in Figure 4.4. In the initial state of ρ_Y, the

gun is not loaded, Suzy is not in possession of the gun, and the turkey is alive. In the second state, Suzy has the gun, and the gun is loaded in the third. In the final state, the turkey is dead the path after the occurrence of ϵ_1, ϵ_2, and ϵ_3.

It is straightforward to verify for the problem $\psi_Y = \langle \theta_Y, \rho_Y, AD \rangle$ that σ_4 is the only transition state of θ_Y in ρ_Y and that ϵ_3 is the causing compound event of $\neg isAlive(turkey)$. The elementary event $shoots(suzy, turkey) \in \epsilon_3$ is a direct cause of $\neg isAlive(turkey)$ because $\neg isAlive(turkey) \in E(shoots(suzy, turkey),$
$\sigma_3)$, as per rule (4.11).

There is more causal information to uncover in this problem. If we want to know why the gun was loaded so that Suzy was able to shoot the turkey, we can use the preconditions of rule (4.12) to formulate a new problem. According to this rule, Suzy cannot shoot the turkey if the gun is not loaded, and so we create the problem $\psi'_Y = \langle \{isLoaded(gun)\}, \rho_Y, AD \rangle$ (using the path and action description of ψ'_Y) and determine that $loads(suzy, gun)$ directly caused the gun to be loaded. Finally, if we want to know why she was able to load the gun in the first place, we use rule (4.15) to create a new problem $\psi''_Y = \langle \{hasGun(suzy)\}, \rho_Y, AD \rangle$ to determine that $handsGunTo(tommy, suzy)$ directly caused Suzy to have the gun. Table 4.2 summarizes the causal explanations for each of the outcomes we considered in this problem. Each row of the first column gives the outcome of interest of the problems ψ_Y, ψ'_Y, and ψ''_Y, respectively. The remaining three columns provide information about the transition states as well as direct and indirect causes. It is easy to see that there are no indirect causes for this example. Note that although in this case we manually constructed our new problems as to uncover additional information about the causal mechanism, it is possible to define sets of supporting events in a scenario that helped to "set the state" for the outcome to be caused, as we explored in an earlier version of the framework that did not support concurrent events [53].

With this example, we demonstrated that our framework can be used to reach a conclusion about the Yale Shooting Problem that appears to be in line with human intuition.

Table 4.2: Overview of outcomes $\{\neg isAlive(turkey)\}$, $\{isLoaded(gun)\}$, and $\{hasGun(suzy)\}$ in transition states σ_4,σ_3, and σ_2 in ρ_Y, respectively.

Outcome of Interest	*State*	*Direct Causes*	*Indirect Causes*
$\{\neg isAlive(turkey)\}$	σ_4	$shoots(suzy, turkey) \in \epsilon_3$	–
$\{isLoaded(gun)\}$	σ_3	$loads(suzy, gun) \in \epsilon_2$	–
$\{hasGun(suzy)\}$	σ_2	$handsGunTo(tommy, suzy) \in \epsilon_1$	–

4.3 Firing Squad Problem

In this section we present a problem based on Pearl's well-known firing squad example [54] to demonstrate that we can represent and identify contributory causes without using laws that explicitly describe the direct effects of compound events. The scenario is as follows:

> *A captain will orders his team of three riflemen to execute two prisoners. The riflemen always hit their targets, but with varying accuracy. The first rifleman is very skilled and if he fires alone, the prisoner will die. The second and third riflemen are less skilled and as a result they must shoot together to ensure a prisoner's death. The captain orders the execution of the first prisoner by rifleman 1, he fires, and prisoner 1 is dead. At the same time that rifleman 1 fires, the captain riflemen 2 and 3 to execute the second prisoner, they fire, and prisoner 2 is dead. What is the cause of the prisoners deaths?*

The extended firing squad example tuple is given by $\Psi_F = \langle \theta_F, v_F, AD_F \rangle$ where the outcome of interest is $\theta_F = \{\neg isAlive(p1), \neg isAlive(p2)\}$ and the sequence of events corresponding to the firing squad is given by $v_F = \{\epsilon_1, \epsilon_2\}$ where $\epsilon_1 = \{ordersExecution(c, p1)\}$, $\epsilon_2 = \{firesAt(r1, p1), ordersExecution(c, p2)\}$, and $\epsilon_2 = \{firesAt(r2, p2), firesAt(r3, p2)\}$. The action description AD_F characterizes the events of the firing squad domain:

$$\begin{cases} ordersExecution(c, p1) \textbf{ causes } orderedToFire(r1) & (4.17) \\ ordersExecution(c, p2) \textbf{ causes } orderedToFire(r2), & (4.18) \\ \qquad\qquad\qquad\qquad\qquad\qquad orderedToFire(r3) \end{cases}$$

$$\mathit{firesAt}(R, p) \textbf{ causes } \mathit{hitBy}(p, R) \tag{4.19}$$

$$\mathit{firesAt}(R, p) \textbf{ impossible\_if } \neg \mathit{orderedToFire}(R) \tag{4.20}$$

$$\neg \mathit{isAlive}(p1) \textbf{ if } \mathit{hitBy}(p, r1) \tag{4.21}$$

$$\neg \mathit{isAlive}(p2) \textbf{ if } \mathit{hitBy}(p, r2), \mathit{hitBy}(p, r3) \tag{4.22}$$

Law (4.17) states that if the captain c orders the execution of prisoner $p1$, then rifleman $r1$ is ordered to fire. Similarly, law (4.18) states that riflemen $r1$ and $r2$ are ordered to fire if c orders the execution of prisoner $p2$. Law (4.19) tells us that a rifleman (the shorthand R refers to any rifleman) firing at the prisoner causes the prisoner to be hit by that rifleman. Law (4.20) states that a rifleman cannot fire at a prisoner unless the captain has ordered the execution. Finally, laws (4.21) and (4.22) capture the requirements from the scenario that one shot from rifleman $r1$ is sufficient to kill prisoner $p1$ and simultaneous shots by riflemen $r2$ and $r3$ are sufficient to kill prisoner $p2$, respectively. In this model of the scenario, we state that firing at the prisoner directly causes the prisoner to be hit by the rifleman, which in turn results in his death.

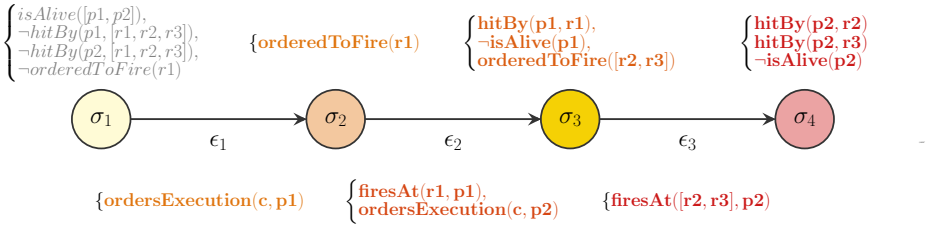

Figure 4.5: Path $\rho_F \in S(\Psi_F)$ is a representation of the firing squad scenario.

Consider the scenario path $\rho_F \in S(\Psi_F)$, graphically represented[2] in Figure 4.5. In the initial state, both prisoners are alive, represented by the literal $\mathit{isAlive}([p1, p2])$. In this state, they have not been hit by any riflemen, and the riflemen have not yet been ordered to fire, represented in σ_1 by literal $\neg \mathit{orderedToFireAt}([r1, r2, r3])$. Finally, the prisoners are both alive in σ_1, represented by the literal $\mathit{isAlive}([p1, p2])$, and they have not yet been hit by any rifleman, represented by $\neg \mathit{hitBy}(p1, [r1, r2, r3])$ and $\neg \mathit{hitBy}(p2, [r1, r2, r3])$. The compound events ϵ_1, ϵ_2, and ϵ_3 in path

[2]For clarity of presentation, we use shorthand $[r1, r2, r3]$ and $[r2, r3]$ for all fluents and events referring to more than one rifleman. We will sometimes use similar shorthand when referring to both prisoners.

ρ_F capture the events of the scenario. The prisoners are no longer alive in state σ_3.

It is straightforward to verify for the problem $\psi_F = \langle \theta_F, \rho_F, AD \rangle$ that σ_4 is the only transition state of the outcome θ_F in ρ_F, that the literal $\neg isAlive(p1) \in \theta_F$ holds in σ_3 as an effect of compound event ϵ_2, and that the literal $\neg isAlive(p2) \in \theta_F$ holds in σ_4 as an effect of compound event ϵ_3. The compound event $\epsilon = \{\mathit{firesAt}(r1, p1)\}$ in ϵ_2 is an indirect cause of $\neg isAlive(p1)$ for σ_3 because there exists static chain with no extraneous events linking its direct effects to $\neg isAlive(p1)$ via the dynamic law (4.19), where $R = r1$, and the state constraint (4.21). Similarly, $\epsilon' = \{\mathit{firesAt}(r2, p), \mathit{firesAt}(r3, p)\}$ in ϵ_2 is an indirect cause of $\neg isAlive(p2)$ for σ_4 via the dynamic laws (4.19), where $R = r2$ and $R = r3$, and the state constraint (4.22). Note that although there is a static chain $\chi(\{firesAt(r1, p1),$
$ordersExecution(c, p2)\}, \neg isAlive(p1), \sigma_3)$, the subset $\{firesAt(r1, p1),$
$ordersExecution(c, p2)\}$ is not an indirect cause because the captain ordering the execution of the second prisoner had no bearing on the death of the first. In the first three sections of this chapter, we have demonstrated that the framework is capable of identifying actual causation in interesting extensions of well-known examples from the literature.

4.4 Self-Driving Car Problem

In this section, we present a example based on distributed system architecture models of self-driving cars (e.g. [55]) in which the components required to operate a car are broken into distinct submodules that communicate with one another to make driving decisions. For safety, a fully autonomous (without human control) driving system with this type of architecture is likely to have multiple backup modules and systems that can take over specific driving-related tasks in case the primary module is somehow compromised. In this example, we assume that a combination of necessary modules are communicating with one another behind the scenes to operate the car. Moreover, we expect that there are multiple types of each required module, which can be swapped in and out at any time as needed by some unseen mechanism (e.g. on-board system reconfiguration software). We will rely on the active modules only to report the actions they take so that we can reason about how these actions affect the state of the car over time. In this section we will provide

preliminary information about a hypothetical self-driving car and will then present a crash scenario as a problem ψ_S and leverage the reasoning framework to identify a set of causal explanations that will indicate which modules may be to blame for the crash.

4.4.1 Example Preliminaries

A self-driving car in this example is represented by $S = \langle \rho_S, AD_S \rangle$, where ρ_S is a path in $\tau(AD_S)$ representing the evolution of S's state. The car drives in a single direction and is able to accelerate, decelerate, and come to a full stop. Obstacles may be placed in its path, some of which will cause the car to crash if it accelerates towards such an obstacle.

The automated driving system of S must have exactly one type of obstacle detection module (types o_1 and o_2), one type of decision making module (types d_1 and d_2), and one type of control module (types c_1 and c_2) active at a given point in time. The configuration of active modules can change in the background without S's knowledge. An obstacle detection module o must report a detected obstacle of type a, b, or c, denoted by elementary events $r(obst(a), o)$, $r(obst(b), o)$, and $r(obst(c), o)$, respectively. A decision making module d can report a decision of type $accel$, $decel$, and $stop$, represented by elementary events $r(decis(accel), d)$, $r(decis(decel), d)$, and $r(decis(stop), d)$, respectively, if it makes a driving decision. Both obstacle detection and decision making modules, generalized as m for this discussion, must report when they are analyzing data, represented by elementary event $r(analyze, m)$. Finally, a control module c must report acting on one of the three types of decisions, given by the elementary events $r(actOn(accel), c)$, $r(actOn(decel), c)$, and $r(actOn(stop), c)$. Modules can act simultaneously and each compound event ϵ in ρ_S represents a collection of actions executed by modules of S.

The knowledge of events in this domain are characterized by the action description AD_S containing the following laws (4.23) through (4.28).

$$r(obst(B), O) \textbf{ causes } known(obst(B)) \textbf{ if } obstMod(O) \tag{4.23}$$

$$r(decis(A), D) \textbf{ causes } mustAct(A) \textbf{ if } decMod(D) \tag{4.24}$$

$$r(actOn(A), C) \textbf{ impossible\_if } \neg mustAct(A) \tag{4.25}$$

$$r(actOn(A), C) \textbf{ causes } \neg mustAct(A) \textbf{ if } ctrlMod(C) \tag{4.26}$$

$$r(actOn(accel), C) \textbf{ causes } infer(crash) \textbf{ if } known(obst(a)) \tag{4.27}$$

$$error(crash) \textbf{ if } infer(crash) \tag{4.28}$$

Law (4.23) states that if an obstacle detection module reports an obstacle, then the car knows about it, which is represented by the literal $known(obst(B))$. Law (4.24) states that if a decision module reports a decision A, then the decision A must be acted upon, represented by the literal $mustAct(A)$. Law (4.25) tells us that a control module will not report an action unless there is knowledge that a corresponding decision that must be acted upon. AD_S further specifies that a decision only needs to be acted upon once via law (4.26). Notice that the inclusion of predicates $obstMod$, $decMod$, and $ctrlMod$ in laws (4.23), (4.25), (4.26), and (4.27) can be viewed as something of a layer of security that ensures only reports from known modules are able to affect the state of the car.

Law (4.27) clarifies the earlier presented knowledge that accelerating the car with knowledge of an obstacle of type a will result in a crash. Finally, law (4.28) tells us that the error message $error(crash)$ will become true if we can infer that a crash has occurred via law (4.27).

4.4.2 Crash Scenario and Explanations

A self-driving car $S = \langle \rho_S, AD_S \rangle$ indicates that it has crashed with the message $error(crash)$. We formulate the problem $\psi_S = \langle \theta_S, \rho_S, AD_S \rangle$, where $\theta_S = \{error(crash)\}$ in order to learn about why S believes that a crash occurred.

Path Description. The crash scenario according to S is given by the path ρ_Y, depicted in Figure 4.6. In the figure, the initial state is fully specified and the subsequent states are represented by the literals that have changed as a result of their respective transitions. The initial state of the path includes a fluent $obstMod(O)$, $decMod(D)$, and $ctrlMod(C)$ for each type of respective module (e.g. o_1, d_1, and c_1). All remaining literals in the initial state, depicted above state σ_1 in the figure, are initially

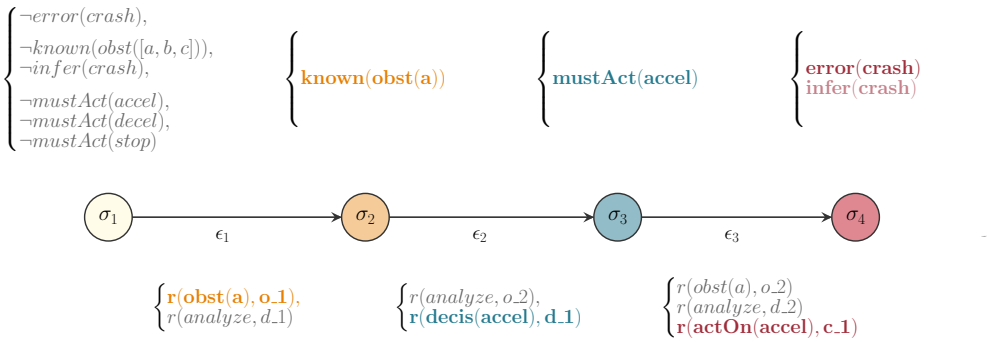

Figure 4.6: Path ρ_S is a representation of car S's crash scenario.

false. This means that there is not yet an error message or inference about a crash. Moreover, the fact that there is no known obstacle is given by the shorthand $\neg known(obst([a, b, c]))$, representing three distinct three distinct literals of the form $\neg known(obst(OB))$ where $OB \in \{a, b, c\}$.

The car is not initially obligated to perform any action, possibly indicating that S is stopped in state σ_1. In compound event ϵ_1, module o_1 reports an obstacle of type a and module d_1 reports that it is analyzing data. In compound event ϵ_2, module o_1 has apparently been swapped out for o_2, which reports that it is analyzing data. In the same compound event, module d_1 reports a decision to accelerate. Finally, in compound event ϵ_3, module o_2 reports an obstacle of type a, module d_1 has been swapped out for d_2, which reports that it is currently analyzing data, and module c_1 reports that it is acting on the decision to accelerate. The states of ρ_S can be inferred using AD_S, and the figure depicts the literals that were changed in each state by the events of ρ_S. Namely, o_1's report added knowledge of an obstacle of type a in σ_1, d_1's report in ϵ_2 obligated the car to accelerate, and the control module c_1's report about acting upon the decision to accelerate caused S to infer the crash in σ_4, which in turn caused $error(message)$ to hold in the same state.

Explaining the Crash Error Message. It is straightforward to verify via ρ_S that σ_4 is the only state of ρ_S in which $error(crash)$ holds. Compound event ϵ_3 is clearly the causing compound event of $error(crash)$ because σ_4 is the only transition state of this literal. The compound event $\{r(actOn(accel), c\_1)\}$ of ϵ_3 is a indirect cause of $error(crash)$ in σ_4 due to laws (4.27) and (4.28). In other words, this causal explanation tells us that the act of accelerating in state σ_3 caused S to infer,

using knowledge of $known(obst(a))$ that there was a crash, which indirectly caused the error.

Identifying Additional Modules to Blame. While it is useful to learn which event led to the crash error message, there is not yet enough information to assign blame to a module. A next step that can be taken is to identify which module reported the decision to accelerate, resulting the system needing to act on that decision. It is straightforward to verify in this case that d_1's act of reporting the decision directly obligated the car to accelerate because $mustAct(accel) \in E(r(decis(accel), d_1))$. It is also straightforward to reason that o_1 was the obstacle detection module that reported the obstacle of type a in ϵ_2.

In this example we have demonstrated that the framework can be used to reason to varying depths about the nature of an error message in a theoretical distributed system. Although the example is relatively simple compared to real-world distributed systems, we show that it may be possible to add a reasoning layer over a system of independently acting parts of a complex system in order to pin-point the source of unexpected behavior of the system as a whole.

Chapter 5: Implementation

In this chapter we present an approach to automating the task of explaining actual cause using ASP. We first discuss the translation of a problem $\psi = \langle \theta, \rho, AD \rangle$. Next, we present the encoding of the semantics of $\mathcal{AL}$, adapted from [43]. We then proceed to encode the definitions of the framework as ASP rules. Following that, we present soundness and completeness results about the correctness of the programs encoding the definition of direct cause and the improved definition of indirect cause. Finally, we will present ASP translations of the extended Yale Shooting Problem and the self-driving car problem from Chapter 4

5.1 Implementation Details

5.1.1 Problem Translation

We begin by encoding the elements of a problem $\psi = \langle \theta, \rho, AD \rangle$. The set $\alpha(\theta)$ contains a fact $outcome(theta)$ as well as facts $inOutcome(l, theta)$, $olit(l)$, and $inOutcome(l, olit(l))$ for every $l \in \theta$. We use the $olit(l)$ notation to denote the outcome coinciding with the singleton $\{l\}$.

The elements of a path $\rho = \langle \sigma_1, \epsilon_1, \sigma_2, \ldots, \epsilon_k, \sigma_{k+1} \rangle$ are represented by the sets $\alpha(\rho)$. The set $\alpha(\rho)$ contains a fact $occurs(e, i)$ for every $e \in \epsilon_i$ and a fact $holds(l, i)$ for each literal $l \in \sigma_i$ where $1 \geq i < k + 1$. The set also contains facts $event(e)$ and $fluent(f)$ for each $e \in \mathcal{E}$ and $f \in \mathcal{F}$, respectively. Next, for every compound event ϵ_i in ρ, the set $\alpha(\rho)$ also contains facts according to

$$\bigcup_{k \in \mathcal{P}(\epsilon_i) \setminus \emptyset} \{subset(\lambda(k, i), i)\} \cup (\bigcup_{e \in k} \{inSubset(e, \lambda(k, i))\})$$

where $\mathcal{P}(\epsilon_i)$ denotes the powerset of ϵ_i, and $\lambda(k, i)$ is a unique identifier for the translation of subset k of ϵ_i. This encoding will be helpful when we are considering subsets of compound events as indirect causes.

The set $\alpha(\rho)$ finally contains a fact $step(i)$ for every state σ_i in ρ. The set $\alpha(\rho)$ represents all knowledge of the path ρ needed to reason for direct and indirect causes.

The set $\alpha(AD)$ is the translation of the action description AD.

1. $\alpha(d : e$ **causes** l_0 **if** $l_1, \ldots, l_n)$ is the collection of atoms

$$d\_law(d), head(d, l_0), event(d, e),$$

$$prec(d, 1, l_1), \ldots, prec(d, n, l_n), prec(d, n+1, nil).$$

2. $\alpha(s : l_0$ **if** $l_1, \ldots, l_n)$ is the collection of atoms

$$s\_law(s), head(s, l_0),$$

$$prec(s, 1, l_1), \ldots, prec(s, n, l_n), prec(s, n+1, nil).$$

3. $\alpha(\iota : e$ **impossible_if** $l_1, \ldots, l_n)$ is the collection of atoms

$$i\_law(\iota), event(\iota, e),$$

$$prec(\iota, 1, l_1), \ldots, prec(\iota, n, l_n), prec(\iota, n+1, nil).$$

The set $\alpha(\psi) = \alpha(\theta) \cup \alpha(\rho) \cup \alpha(AD)$ is the complete ASP encoding of the problem ψ. Next, we present the encoding of the semantics of $\mathcal{AL}$.

5.1.2 Semantics of Action Language $\mathcal{AL}$

The rules of program $\Pi_{\mathcal{AL}}$ capture the semantics of $\mathcal{AL}$ and are adapted from [43]:

$$\begin{aligned} holds(L, I2) \leftarrow \quad & step(I1),\ step(I2),\ next(I1, I2) \\ & d\_law(D),\ head(D, L),\ prec\_h(D, I) \\ & event(D, E),\ occurs(E, I1). \end{aligned} \tag{5.1}$$

$$\begin{aligned} holds(L, I) \leftarrow \quad & step(I), \\ & s\_law(S),\ head(S, L),\ prec\_h(S, I). \end{aligned} \tag{5.2}$$

$$\leftarrow \quad step(I), \tag{5.3}$$
$$i\_law(IM),\ prec\_h(IM, I),$$
$$event(IM, E),\ occurs(E, I).$$

$$prec\_h(R, I) \leftarrow \quad step(I), \tag{5.4}$$
$$all\_h(R, 1, I).$$

$$all\_h(R, N, I) \leftarrow \quad prec(R, N, nil). \tag{5.5}$$

$$all\_h(R, N, P) \leftarrow \quad step(I) \tag{5.6}$$
$$prec(R, N, P),$$
$$holds(P, I),$$
$$all\_h(R, N + 1, I).$$

$$holds(L, I2) \leftarrow \quad holds(L, I1),\ \text{not}\ holds(\overline{L}, I). \tag{5.7}$$

$$\leftarrow \quad holds(L, I),\ holds(\overline{L}, I). \tag{5.8}$$

$$next(I1, I2) \leftarrow \quad step(I1),\ step(I2), \tag{5.9}$$
$$I2 = I1 + 1.$$

D, S, IM are variables for the names of $\mathcal{AL}$ laws, E and L are variables for the names of events and fluent literals respectively, and I and N are integers. Rules (5.1) and (5.2) in Π describe the effects of dynamic laws and and state constraints of AD. Rule (5.3) constrains when an event cannot occur according to the executability conditions in AD. Relation $prec\_h(R, I)$ defined by(5.4) of Π tells us that all of the preconditions of a law R in $\alpha(AD)$ are satisfied at step I. This relation is defined via an auxiliary relation $all\_h(R, N, I)$ (rules (5.5) and (5.6)), which holds if the preconditions l_1,...,l_m of R are satisfied at step I. Here, $l_1, \ldots, l_m$ refer to the ordering of conditions of r according to $\alpha(AD)$. We use the identifier R to enable re-use of rules (5.4), (5.5), and (5.6) to reason about the preconditions of dynamic law, state constraints, and executability conditions. Rule (5.7) is the inertia axiom [39] and rule (5.8) removes inconsistent states in the reasoning process. In rules

(5.7) and (5.8), we denote by $\overline{L}$ the complement of L and so given a literal l, $\bar{l} = \neg l$ and $\overline{\neg l} = l$. We explicitly represent subsequent steps by means of the rule (5.9).

5.1.3 Transition States and Causing Compound Events

The rules in set Π_T characterize a transition state σ_j of θ in path ρ.

$$\begin{aligned} transitionState(OC, J') \leftarrow \quad & step(J),\ step(J'), \\ & next(J, J'), outcome(OC), \\ & \neg ocSat(OC, J), \\ & \text{not } \neg ocSat(OC, J') \end{aligned} \tag{5.10}$$

$$\begin{aligned} \neg ocSat(OC, J) \leftarrow \quad & step(J), \\ & inOutcome(OC, L), \\ & holds(\overline{L}, J). \end{aligned} \tag{5.11}$$

Rule (5.10) identifies a transition state J' for outcome OC as one in which OC is satisfied and was not at J. Rule (5.11) describes when an outcome OC is not satisfied at a given step J.

The rules of program Π_C describe possibly causing compound events and causing compound events of a literal L that holds at step I.

$$\begin{aligned} pcce(I, L, J) \leftarrow \quad & step(I),\ step(J), next(I, I'), \\ & transitionState(olit(L), I'), \\ & I < J, \\ & transitionState(theta, J). \end{aligned} \tag{5.12}$$

(5.13)

$$\neg cce(I, L, J) \leftarrow \quad pcce(I, L, J), \tag{5.14}$$
$$pcce(I', L, J),$$
$$I < I'.$$

$$cce(I, L, J) \leftarrow \quad pcce(I, L, J), \tag{5.15}$$
$$\text{not } \neg cce(I, L, J).$$

Rule (5.12) corresponds to Definition 5 and tells us that a step I is a *possibly causing compound event* of literal L holding in J if, first, it occurs prior to step J. Next, I' must be a transition state of $outcome(olit(L))$. The last line of (5.12), $transitionStep(theta, J)$ is the key to linking the literal we are explaining to the outcome of interest. This line ensures that any possible causing step we are considering is with respect to a transition state of the $outcome(theta)$.

Rule (5.14) corresponds to condition 3 of Definition 6 and (5.15) is a straightforward rule stating that a possible causing step I of L in step J is a causing step if we have no reason to believe that it is *not* a causing step.

5.1.4 Direct Cause

Here we present characterization of the definition of direct cause. The rules of Π_D describe when an event that occurred at causing step I has directly caused L to hold in step I. Π_D contains the rules corresponding to the definition of a direct cause.

$$directEffect(L, E, I) \leftarrow \quad step(I), \tag{5.16}$$
$$d\_law(D), event(D, E), occurs(E, I),$$
$$prec\_h(D, I), head(D, L).$$

$$directCause(E, I, L, J) \leftarrow \quad cce(I, L, J), \tag{5.17}$$
$$directEffect(L, E, I).$$

Rule (5.16) leverages rule $prec\_h(D, I)$ of $\Pi_{\mathcal{AL}}$ to identify when the literal L will be caused as a direct effect of an event occurring at step I. Rule (5.17) states that E occurring at I is a direct cause of L holding at step J if I is a causing step of L in J and L is a direct effect of E as per rule (5.16).

5.1.5 Simple Notion of Indirect Cause

The program Π_{I_S} contains the following rules (5.18) through (5.30), which are used to identify indirect causation for Definition 8. Given a causing step I,we are interested in identifying any subset of events that occurred at I and caused the literal under consideration to hold indirectly. The following rule describes possible indirect causes.

$$\begin{aligned} possIC(C, I, L, J) \leftarrow \quad & subset(C), \\ & causingStep(I, L, J), \\ & \text{not } \neg occAt(C, I), \\ & \text{not } dirEffSubset(L, C, I). \end{aligned} \tag{5.18}$$

Rule (5.18) says that a possible indirect cause is a subset C such that I is a causing step for the literal L holding at step J and we have no reason to believe that any event in C did not occur at step I or that any events in the subset caused the literal directly. The next rule ensures that the compound event under consideration actually occurs as a whole at step I.

$$\begin{aligned} \neg occAt(C, I) \leftarrow \quad & step(I), \\ & inSubset(E, C), \\ & \text{not } occurs(E, I). \end{aligned} \tag{5.19}$$

We determine when a literal L is a direct effect of at least one elementary event of C using the following rule:

$$dirEffSubset(L, C, I) \leftarrow \quad inSubset(E, C), \quad directEffect(L, E, I). \tag{5.20}$$

Rule (5.20) leverages rule (5.16) to determine when a literal L is a direct effect of a subset of events C. Recall that condition 2 of Definition 8 states that if $\epsilon' \in \epsilon_i$ is an indirect cause of l, then a transition t' must exist in $\tau(AD)$ such that if ϵ' occurring by itself in σ_i results in a transition state of $\{l\}$. Given a possible indirect cause $possIC(C, I, L, J)$, this reasoning can be accomplished in ASP by creating a hypothetical sequence whose initial state and events coincide with those of interest from the sequence of steps, namely step I and the events that occur at I. We refer to this check as the *hypothetical reasoning test*.

$$next(\mu(C, I), \mu'(C, I)) \leftarrow \quad possIC(C, I, L, J), \tag{5.21}$$

$$hstep(\mu(C, I)) \leftarrow \quad next(\mu(C, I), \mu'(C, I)). \tag{5.22}$$

$$hstep(\mu'(C, I)) \leftarrow \quad next(\mu(C, I), \mu'(C, I)). \tag{5.23}$$

$$step(I) \leftarrow \quad hstep(I). \tag{5.24}$$

Rules (5.21), (5.22), and (5.23) establish an ordering for *hypothetical steps*, or h-steps. Rule (5.24) states that an h-step is a kind of step. The following rules ensure the coincidence of the hypothetical sequence with the relevant step and event(s) of $\alpha(\rho)$.

$$holds(L, \mu(C, I)) \leftarrow \quad holds(L, I), \tag{5.25}$$
$$possIC(C, I, L, J),$$

$$occurs(E, \mu(C, I)) \leftarrow \tag{5.26}$$
$$possIC(C, I, L, J), \tag{5.27}$$
$$inSubset(E, C),$$

In order to indicate that subset C has passed the hypothetical reasoning test for causing L in step I, we add the following rule which we will use later to describe indirect causation.

$$hypotheticalPass(C, I, L, J) \leftarrow \quad possIC(C, I, L, J), outcome(olit(L)), \tag{5.28}$$
$$transitionStep(olit(L), \mu(C, I)).$$

Next, rule (5.29) ensures that the candidate subset C is a smallest subset for which $possIC(C, I, L, J)$ holds by comparing C's cardinality against all other possible indirect causes $possIC(C', I, L, J)$ where $C \neq C'$.

$$\begin{aligned} \neg smallest(C,L,I) \leftarrow \quad & possIC(C,I,L,J), \\ & subset(C'), C \neq C', \\ & possIC(C',I,L,J), \\ & \#count\{E : inSubset(E,C)\} = X, \\ & \#count\{E : inSubset(E,C')\} = X', \\ & X > X'. \end{aligned} \tag{5.29}$$

Finally, we can say that a subset C is an indirect cause if it is the smallest one that passes the hypothetical reasoning test.

$$\begin{aligned} indirectCause(C,I,L,J) \leftarrow \quad & hypotheticalPass(C,I,L,J), \\ & \text{not } \neg smallest(C,L,I). \end{aligned} \tag{5.30}$$

The connection between direct causes and the above programs is given later in this chapter by Theorem 1. The connection between the above programs and the simple notion of indirect cause is given by the following proposition.

Proposition 6. *Let $\psi = \langle \theta, \rho, AD \rangle$ be a problem, ϵ_i be a compound event in ρ, σ_j be a transition state of θ in ρ, l be a literal in θ, and Π_ψ be a problem translation of ψ. Given the program $\Pi_{indirect_S} = \Pi_\psi \cup \Pi_{\mathcal{AL}} \cup \Pi_T \cup \Pi_C \cup \Pi_{I_S}$, compound event $\epsilon \in \epsilon_i$ is a indirect cause of l holding in σ_j if-and-only-if the atoms $indirectCause(c,i,l,j)$ and $subset(c,i)$ are in the answer set of $\Pi_{indirect}$, as well as an atom $inSubset(e,c)$ for every $e \in \epsilon$.*

5.1.6 Improved Definition of Indirect Cause

The program Π_I contains the following rules (5.31) through (5.42) which identify indirect causation for Definition 5.42. Given a step I, the following rule leverages the ASP translation of the seman-

tics $\mathcal{AL}$ of and allows us to determine which literal(s) of a state constraint's preconditions were preserved in I by inertia:

$$preservedPrec(S, L, I') \leftarrow \quad s\_law(S), prec\_h(S, I'), \\ prec(S, X, L), holds(L, I), next(I, I'). \tag{5.31}$$

The next rule requires that when a literal in S's preconditions was caused to hold directly by a subset C of the events that occurred in step I.

$$directlyCausedPrec(S, C, L, I') \leftarrow \quad s\_law(S), prec\_h(S, I'), inSubset(E, C), \\ prec(S, X, L), directEffect(L, E, I), \\ holds(\overline{L}, I), next(I, I'). \tag{5.32}$$

Note that line 3 in $directlyCaused(S, C, L, I')$ requires that a literal that is directly caused must not have held in the previous state, in correspondence with our condition that inertia is prioritized over direct causation. The next rule characterizes the first link of a static chain:

$$\begin{aligned} gamma(1, S, C, L, I') \leftarrow \quad & subset(C, I), holds(L, I'), s\_law(S), \\ & prec\_h(S, I'), next(I, I'), \\ & \#count\{L1 : preservedPrec(S, L1, I')\} = N1, \\ & \#count\{L2 : directlyCausedPrec(S, C, L2, I')\} = N2, \\ & \qquad\qquad\qquad\qquad\qquad\qquad N2 > 0, \\ & \#count\{L3 : prec(S, X, L3), L3! = nil\} = N3, \\ & N3 = N1 + N2. \end{aligned} \tag{5.33}$$

Rule (5.33) tells us when a state constraint S belongs in the first link of a chain. We use the $\#count$ aggregate to compare the number of preserved preconditions of S and the number of preconditions directly caused by C with the total number of preconditions for S (excepting the "nil" precondition), in accordance with Condition 1 of Definition 10. In the third line of this rule, we require that the set of directly caused preconditions of S in the first link is non-empty by constraining this sum to be a value greater than zero. Preventing the overlap of preserved and directly caused preconditions of S (see rule (5.32)) allows us to use the inequality on 7th line of this rule to satisfy condition 1 of Definition 9. We use a similar approach to characterize subsequent links of a static chain.

$$
\begin{aligned}
gamma(G+1,S,C,L,I') \leftarrow\ & subset(C,I), holds(L,I'), s\_law(S), prec\_h(S,I'), && (5.34)\\
& next(I,I'), gamma(G,X,C,L,I'),\\
& \#count\{L1 : preservedPrec(S,L1,I')\} && = N1,\\
& \#count\{L2 : directlyCausedPrec(S,C,L2,I')\} && = N2,\\
& \#count\{L3 : causedByGamma(S,L3,G,C,L,I')\} && = N3,\\
& && N3 > 0,\\
& \#count\{L4 : causedByGamma(S,L4,G',C,L,I'),\\
& && G' <= (G-1)\} = N4,\\
& \#count\{L5 : prec(S,X,L5), L5\,! = nil\} = N5,\\
& N5 = N1 + N2 + N3 + N4, && (5.35)\\
& \text{not } consequenceOfGamma(L,G,C,L,I').
\end{aligned}
$$

Here we aim to characterize link $G+1$ of the chain in terms of the earlier links in the chain, direct effects of events, and inertia. Recall that in condition 2 of Definition 9, the set of preserved and directly caused preconditions can be empty. However, the definition does require that the number

of preconditions of S made to hold as a consequence of the immediately previous link is greater than zero for any link following the first link in a chain. Before continuing the discussion of (5.34), we will enforce this condition with some auxiliary rules:

$$consequenceOfGamma(L', G, C, L, I') \leftarrow \quad gamma(G, S, C, L, I'), head(S, L'). \tag{5.36}$$

It is easy to see that this rule characterizes literals that belong to the set of consequences $C(\gamma_g)$ of a link γ_g. The next rule characterizes preconditions of a particular state constraint have been caused by a given link.

$$\begin{aligned} causedByGamma(S, L', G, C, L, I') \leftarrow \quad & s\_law(S), prec\_h(S, I'), \\ & prec(S, X, L'), \\ & consequenceOfGamma(L', G, C, L, I'). \end{aligned} \tag{5.37}$$

Returning to rule (5.34), the 5th line counts the number of preconditions of a candidate S for the $G+1$-th link caused by the immediately previous link J with the constraint that the value of this sum is greater than zero. The 6th line also leverages (5.37) to count the number of preconditions of S satisfied by links that appear earlier in the chain than the immediately previous link. Finally, in the 8th line we require that the literal L is not a consequence of the G-th link in accordance with condition 2 of Definition 9.

For the simplicity of the presentation, we have given the encoding for the case of condition 2 of Definition 9 for which $C(\{\gamma_1, \ldots, \gamma_{g-2}\}) \cap C(\gamma_{g-1}) = \emptyset$. Extending the program to the case where there is overlap is trivially accomplished with the addition of a rule such as

$$overlap(G, L', C, L, I) \leftarrow causedByGamma(S, L', G, C, L, I'),$$
$$causedByGamma(S, L', G', C, L, I'),$$
$$G' < G.$$

along with the addition of a suitable $\#count$ aggregate to rule (5.33). Herein we refer to the case without overlap as a *no overlap case*.

The following rule characterizes the existence of a chain $chain(C, L, I)$

$$chain(C, L, I') \leftarrow gamma(G, S, C, L, I'), \quad (5.38)$$
$$consequenceOfGamma(L, G, C, L, I').$$

It is now easy to see that these rules correspond to the conditions of the definition of a static chain. Once a chain is found, it needs to be tested to determine whether or not it is actually an indirect cause of the literal in question as per Definition 10 of indirect cause. If the chain $chain(C, L, I)$ exists, then we claim that Condition 1 of the definition is satisfied for C.

It remains to rule out chains whose subsets have extraneous events, that is to say that C does not contain any events whose direct effects do not appear in the preconditions of any link of the chain under consideration as per the definition of indirect cause. The following rule characterizes the set of preconditions for link G:

$$preconditionOfGamma(P, G, C, L, I') \leftarrow gamma(G, S, C, L, I'), \quad (5.39)$$
$$prec(S, X, P), P! = nil.$$

Rule (5.40) leverages the previous rule to identify when an event E belonging to a subset C has

directly contributed to the preconditions of any link Y in a chain linking C and L at step I':

$$\begin{aligned} contributed(E, P, C, L, I') \leftarrow \quad & preconditionOfGamma(P, Y, C, L, I'), \\ & inSubset(E, C), directEffect(P, E, I), \\ & next(I, I'). \end{aligned} \tag{5.40}$$

Rule (5.41) leverages the previous rule to characterize events that did not contribute to a precondition of any link in a static chain under consideration:

$$\begin{aligned} extraEventsInSubset(C, L, I') \leftarrow \quad & chain(C, L, I'), \\ & inSubset(E, C), \\ & \text{not } contributed(E, P, C, L, I'). \end{aligned} \tag{5.41}$$

If there is no reason to believe that an event E has contributed to the preconditions of any link in the chain under consideration, then the event is extraneous and the subset cannot be an indirect cause. Rules (5.39), (5.40), and (5.41) clearly correspond to condition 2 of Definition 10. Finally, rule (5.42) characterizes the improved definition of indirect cause:

$$\begin{aligned} indirectCause(C, I, L, J) \leftarrow \quad & cce(I, L, J), \\ & chain(C, L, I'), next(I, I'), \\ & \text{not } extraEventsInSubset(C, L, I'). \end{aligned} \tag{5.42}$$

The rule states that if there is a causing compound event at step I, then there exists a subset of events C that has occurred at step I, a static chain links C to L in the subsequent step I', and there are no extraneous events in C that have not contributed to the preconditions of the chain, then C is

an indirect cause of L for transition state J of $outcome(theta)$.

5.2 Theoretical Results

In this section, we state and prove the soundness and completeness of the programs for computing direct cause and improved indirect cause. We begin with the statement for Π_{direct}.

Theorem 1. *Let $\psi = \langle \theta, \rho, AD \rangle$ be a problem, ϵ_i be a compound event in ρ, σ_j be a transition state of θ in ρ, l be a literal in θ, and Π_ψ be a problem translation of ψ. Given the program $\Pi_{direct} = \Pi_\psi \cup \Pi_{\mathcal{AL}} \cup \Pi_T \cup \Pi_C \cup \Pi_D$, elementary event $e \in \epsilon_i$ is a direct cause of l holding in σ_j if-and-only-if the atom $directCause(e, i, l, j)$ is in an answer set of Π_{direct}.*

Proof. Left-to-right.

We begin by characterizing an answer set A of Π_{direct} containing an atom $directCause(e, i, l, j)$. The answer set first contains the problem translation set $\alpha(\psi)$.

- Next, A contains an atom of form $transitionState(theta, j^*)$ for every transition state σ_{j^*} of θ in ρ,
- and an atom $transitionState(olit(l^*), i^*)$ for every transition state σ_i^* of the singleton set $\{l^*\}$ for $l^* \in \theta$.
- A also contains an atom $\neg ocSat(theta, s^*)$ for each state σ_{s^*} in ρ such that $\theta \not\subseteq \sigma_s^*$.
- Similarly, for $l^* \in \theta$ where $\{l^*\} \not\subseteq \sigma_{t^*}$ in ρ, A contains an atom $\neg ocSat(olit(l^*), t^*)$.
- Given a literal $l^* \in \theta$ and a transition state σ_{j^*} of θ in ρ, A contains an atom $pcce(i^*, l^*, j^*)$ for every possibly causing compound event ϵ_{i^*} of $\{l^*\}$ for σ_{j^*}.
- Given possibly causing compound events ϵ_{i^*} and $\epsilon_{i'^*}$ of $\{l^*\}$ for σ_{j^*}, A contains an atom $\neg cce(i^*, l^*, j^*)$ when $i^* < i'^*$.
- A also contains an atom $cce(i^*, l^*, j^*)$ for every causing compound event ϵ_i^* of a literal $l^* \in \theta$ for a transition state σ_j^* of θ.

- For every literal $l^* \in E(e^*, i^*)$ where elementary event e^* occurs at i^* in ρ, A also contains an atom $directEffect(l^*, e^*, i^*)$.

- Finally, A contains an atom $directCause(e, i, l, j)$ for every elementary event $e^* \in \epsilon_{i^*}$ in ρ that is a direct cause of l^* for a transition state σ_{j^*} of θ.

Let us show that if e is a direct cause of l holding in σ_j, then A is an answer set of Π_{direct} by proving that A is the minimal set of atoms closed under the rules of the reduct Π^A_{direct}. We give a simplified form of the reduct where the occurrences of atoms of the form $next(i, j)$ have been unfolded into expressions $j = i + 1$. One can easily check that this form of the reduct is equivalent to the form that uses $next(i, j)$. Π^A_{direct} contains:

1. set $\alpha(\psi)$

2. all rules in $\Pi_{\mathcal{AL}}$

3. a rule of the form

$$transitionState(o^*, j^*) \leftarrow step(j'^*), step(j^*),$$
$$j^* = j'^* + 1, \neg ocSat(o^*, j'^*).$$

   where o^* is either the constant $theta$ or a term of the form $olit(l^*)$ (where l^* is a fluent literal), and integers j^* and j'^* such that $\neg ocSat(o^*, j^*) \notin A$.

4. a rule of the form

$$\neg ocSat(o^*, j^*) \leftarrow step(j^*), holds(\overline{l^*}, j^*), inOutcome(o^*, l^*).$$

   where o^* is either the constant $theta$ or a term of the form $olit(l^*)$ where l^* is a fluent literal, and integers j^*.

5. a rule of the form

$$pcce(i^*, l^*, j^*) \leftarrow step(i^*), step(i'^*), i'^* = i^* + 1,$$
$$transitionState(olit(l^*), i'^*), i^* < j^*,$$
$$transitionState(theta, j^*).$$

   for all literals l^* and integers i^*, j^*.

6. a rule of the form

$$\neg cce(i^*, l^*, j^*) \leftarrow pcce(i^*, l^*, j^*),$$
$$pcce(i'^*, l^*, j^*),$$
$$i^* < i'^*.$$

   for all literals l^* and integers i^*, j^*.

7. a rule of the form

$$cce(i^*, l^*, j^*) \leftarrow pcce(i^*, l^*, j^*).$$

   for all literals l^*, and integers i^*, j^* such that $\neg cce(i^*, l^*, j^*) \notin A$.

8. a rule of the form

$$directEffect(l^*, e^*, i^*) \leftarrow step(i^*), d\_law(d^*), event(d^*, e^*)$$
$$occurs(e^*, i^*), prec\_h(d^*, i^*), head(d^*, l^*).$$

   for all literals l^*, integers i^*, events e^*, and dynamic laws d^*.

9. a rule of the form

$$directCause(e^*, i^*, l^*, j^*) \leftarrow cce(i^*, l^*, j^*),$$
$$directEffect(l^*, e^*, i^*).$$

for all literals l^*, integers i^*, events e^*, and dynamic laws d^*.

A is closed under Π^A_{direct}. We will prove it for every rule of the program.

Rules of group [(1)]: obvious.

Rules of group [(2)]: The conclusion follows from Theorem 1 from [43].

Rules of group [(3)]: If the rule is in Π^A_{direct}, then $\neg ocSat(o^*, j^*) \notin A$ and therefore $o^* \subseteq \sigma_{j^*}$ by construction of A. If the body is satisfied in A, then $o^* \not\subseteq \sigma_{j'^*}$ and $j^* = j'^* + 1$ by construction. By Definition 4, if $o^* \not\subseteq \sigma_{j'^*}$ and $o^* \subseteq \sigma_{j^*}$, then σ_{j^*} is a transition state of o^*. Finally, the atom $transitionState(l^*, j^*) \in A$ by construction of A.

Rules of group [(4)]: If the body is satisfied in A, then there exists a literal $l^* \in o^*$ whose complement $\overline{l^*} \in \sigma_{j^*}$ by construction of A. Because σ_{j^*} is consistent, it must be the case that $l^* \notin \sigma_{j^*}$, which can also be written as $\{l^*\} \not\subseteq \sigma_{j^*}$. Finally, we know from construction of A that whenever there exists a literal $l^* \in o^*$ such that $\{l^*\} \not\subseteq \sigma_{j^*}$, the atom $\neg ocSat(o^*, j^*)$ is added to A.

Rules of group [(5)]: If the body is satisfied in A, then $\sigma_{i'^*}$ in ρ is a transition state of $\{l^*\}$, $i'^* = i^* + 1$, σ_{j^*} in ρ is a transition state of θ in ψ, and $i < j$ by construction of A. By Definition 5, ϵ^*_i is a possibly causing compound event of l^* for transition state σ_{j^*} in ρ. Finally, A contains the atom $possiblyCausingCompoundEvent(i^*, l^*, j^*)$ by construction of the set.

Rules of group [(6)]: If the body is satisfied in A, then there exists possibly causing compound

events ϵ_{i^*} and $\epsilon_{i'^*}$ of a literal $l^* \in \sigma_{j^*}$ such that $i^* < i'^*$. The atom $\neg cce(i^*, l^*, j^*)$ is in the set A by construction.

Rules of group [(7)]: If the rule is in Π^A_{direct}, then $\neg cce(i^*, l^*, j^*) \notin A$. If $pcce(i^*, l^*, j^*)$ inA, then ϵ^*_i is a possibly causing compound event of l^* for transition state σ_{j^*} of ρ. Because $\neg cce(i^*, l^*, j^*) \notin A$, there there is no possibly causing compound event $\epsilon_{i'^*}$ in ρ such that $i^* < i'^*$. By Definition 6, ϵ^*_i is a causing compound event of l^* for transition state σ_{j^*} of ρ. By construction of A, A contains an atom $cce(i^*, l^*, j^*)$.

Rules of group [(8)]: If the body is satisfied in A, then there exists an elementary event $e^* \in \epsilon_{i^*}$ of ρ and a dynamic law $d^* : e^*$ **causes** l^* **if** ω^* and $\omega^* \subseteq \sigma_{i^*}$. By the definition of direct effects in $\mathcal{AL}$, l^* is in the set $E(e^*, \sigma^*_i)$. Therefore, A contains the atom $directEffect(l^*, e^*, i^*)$ by construction.

Rules of group [(9)]: If the atoms $cce(i^*, l^*, j^*)$ and $directEffects(l^*, e^*, i^*)$ are in A, then ϵ_{i^*} is a causing compound event of l^* for transition state σ_{j^*} of θ in ρ and $l^* \in E(e^*, \sigma_{i^*})$ for the elementary event $e^* \in \epsilon_{i^*}$. By Definition 7, $e^* \in \epsilon_{i^*}$ is a direct cause of l^* for σ_{j^*}. Finally, $directCause(e^*, i^*, l^*, j^*)$ is in the set A by construction. We have now proven that A is closed under every rule of Π^A_{direct}.

A is the minimal set closed under the rules of Π^A_{direct}. We will prove this by assuming that there exists a set $B \subseteq A$ such that B is closed under the rules of Π^A_{direct}, and by showing that $B = A$. Our approach will be to show that whenever the head of a rule is in A, the body is in B. Then, because the body is in B and from the fact that B is closed under the rules of the program, it follows that the head is also in B. We will demonstrate this for every rule of the program and conclude that A and B contain the same atoms.

First, the set $\alpha(\psi) \in B$ since these are the facts of Π^A_{direct}.

Rules of group [2]: It is possible to apply the Splitting Set Theorem [56, 57] to Π_{direct} so that the bottom of the program corresponds to $\alpha(SD, \Gamma_D)$ from [43]. The restriction of A to the signature of $\alpha(SD, \Gamma_D)$ is an answer set of $\alpha(SD, \Gamma_D)$ by Theorem 1 from [43] and therefore is minimal. This tells us that whenever the literals of the form $holds(\cdot, \cdot)$, $occurs(\cdot, \cdot)$, $step(\cdot)$, $next(\cdot, \cdot)$, $prec\_h(\cdot, \cdot)$, and $all\_h(\cdot, \cdot, \cdot)$ are in A, then must also be in B.

Rules of group [8]: If the head is in A, then by supportedness the body is also in A. By Theorem 1 of [43], we know that the body is also in B because the atoms are formed from the signature of $\alpha(SD, \Gamma_D)$. Because B is closed under the reduct, the head must also be in B.

Rules of group [4]: If the head of the rule is in A, then by supportedness, the body is in A. Because the facts of A and B coincide, the body is also in B. By closedness of B, the head is also in B.

Rules of group [3]: If the head of the rule is in A, then again by supportedness, the body is in A. Because we have already demonstrated for rules of group 4 that literals of form $\neg occSat(\cdot, \cdot)$ coincide in A and B, then the body is also in B. Because B is closed under the reduct, the head must also be in B.

Leveraging this observation about literals of form $transitionState(\cdot, \cdot)$ coinciding in A and B, we can similarly prove that the coincidence property holds for rules of group 5. Finally, the property holds trivially for rules of groups 6,7, and 9.

We have just proven that for every rule r of the reduct, if $head(r)$ belongs to A, then $head(r)$ belongs to B. By the supportedness property, every literal $l \in A$ must be in the head of some rule. Hence, all literals of A also belong to B. Therefore, $A = B$. We have proven that A is the minimal set of atoms closed under the rules of the reduct Π^A_{direct}.

Right-to-left.

Let A be an answer set of Π_{direct}. We will show that when the atom $directCause(e, i, l, j)$ is in A, then $e \in \epsilon_i$ is a direct cause for the literal l holding in the transition state σ_j of θ from ψ.

Because A is an answer set of Π_{direct}, A is also an answer set of the reduct Π^A_{direct}. Let us consider groups 1 through 9 defined earlier as the reduct.

Rules of group (1): obvious.

Rules of group (2): The conclusion follows from Theorem 1 from [43].

Consider first rules of group 9. If the atom $directCause(e^*, i^*, l^*, j^*)$ belongs to A, then there must be at least one rule in group (9) whose body is satisfied by A by the supportedness property. Therefore, the atom $cce(i^*, l^*, j^*)$ and at least one atom $directEffect(l^*, e^*, i^*)$ are in A.

Consider now the rules of group 8. We have already concluded that there exists at least one literal $directEffect(l^*, e^*, i^*)$ must be in A. Hence, there must be at least one rule in group (9) whose body is satisfied by A. From Theorem 1 of [43], we conclude that there is a dynamic law whose preconditions are satisfied in state σ_{i^*} and whose event belongs to ϵ_{i^*}. By definition of the direct effects of elementary event e^* occurring in state σ_{i^*}, l^* belongs to $E(e^*, \sigma_{i^*})$ (and $E(\epsilon_{i^*}, \sigma_{i^*})$).

In the case of rules of group 7, recall that we have earlier concluded that $cce(i^*, l^*, j^*)$ must be in A. By supportedness, $pcce(i^*, l^*, j^*) \in A$ and, by construction of the reduct, $\neg cce(i^*, l^*, j^*) \notin A$. For rules of group 5, we can similarly conclude that literals $transitionState(o^*, i^*)$ and $transitionState(theta, j^*)$ are in A. Moreover, by the rules of group 3, $\neg ocSat(o^*, i'^*)$, and $\neg ocSat(theta, j'^*)$ are in A, while the literals $\neg ocSat(o^*, i^*)$ and $\neg ocSat(theta, j^*)$ are not.

Consider now the rules of group 4. Because we know that $\neg ocSat(o^*, i'^*)$ is in A, then the body of the corresponding rule of this group must be satisfied in A, including the literal $holds(\overline{l^*}, i'^*)$. By the problem translation $\alpha(\psi)$, $\overline{l^*} \in \sigma_{i'^*}$, which can also be written as and $l^* \notin \sigma_{i'^*}$. Also by $\alpha(\psi)$, we know that l^* is a member of outcome o^*, and because $l^* \notin \sigma_{i'^*}$ it must also be true that $o^* \not\subseteq \sigma_{i'^*}$. Therefore, the condition in Definition 4 that outcome $o^* \not\subseteq \sigma_{i'^*}$ is satisfied when considering a possible transition state σ_{i^*} of o^*.

Recall that if a rule of group 3 is in Π^A_{direct}, then the corresponding literal $\neg ocSat(o^*, j^*)$ is not in A. From the problem translation, $step(j^*)$ and $inOutcome(o^*, l^*)$ are facts in A. With these conditions in mind and looking again at rules in group 4, we see that $holds(\overline{l^*}, j^*)$ must not be in A for any l^* such that $inOutcome(l^*, o^*)$, otherwise $\neg ocSat(o^*, j^*)$ would be in A, which would be a contradiction. By the problem translation and the semantics of $\mathcal{AL}$, it must be that $l^* \in \sigma_{j^*}$ for every $l^* \in o^*$, hence $o^* \subseteq \sigma_{j^*}$. From our earlier conclusion and the fact that $\neg ocSat(o^*, j'^*) \in A$, we get that $o^* \not\subseteq \sigma_{j'^*}$, and therefore the conditions of Definition 4 are satisfied for outcome o^* and state σ_{j^*} in ρ, hence σ_{i^*} is a transition state of o^* in ρ. At this point, we have shown that when literal $transitionState(o^*, j^*)$ is in A, then σ_{j^*} is a transition state of o^* in ρ.

Going back to rules of group 5, the literal $pcce(i^*, l^*, j^*)$ is in A when A also contains literals $step(i^*)$, $step(j^*)$, $transitionState(olit(l^*), i'^*)$ and $transitionState(theta, j^*)$ such that $i'^* = i^* + 1$ and $i^* < j^*$. From our earlier conclusions and by the translation of ψ, we know that $\sigma_{i'^*}$ is a transition state of $\{l^*\}$ and σ_{j^*} is a transition state of θ. The conditions of Definition 5 are met in this case and we conclude that the compound event ϵ_{i^*} in ρ is a possibly causing compound event of l^* for the transition state σ_{j^*} of θ whenever $pcce(i^*, l^*, j^*)$ is in A.

Returning to the rules of group 6, the literal $\neg cce(i^*, l^*, j^*) \in A$ when $pcce(i^*, l^*, j^*)$ and $pcce(i'^*, l^*, j^*)$ are in A such that $i^* < i'^*$. From our earlier conclusions, ϵ^*_i and ϵ'^*_i are both possibly causing

compound events of l^* for transition state σ_{j^*} of the outcome o^*. However, the conditions of Definition 6 are violated due to the inequality of $i^* < i'^*$. Therefore, ϵ_i^* cannot be a causing compound event of l^* for the transition state σ_{j^*} of θ whenever $\neg cce(i^*, l^*, j^*) \in A$.

Considering again the rules of group 7, the literal $cce(i^*, l^*, j^*)$ is in A when $pcce(i^*, l^*, j^*)$ is in A and $\neg cce(i^*, l^*, j^*)$ is not in A. We know from our earlier conclusions that ϵ_i^* is a possibly causing compound event of l^* for transition state σ_{j^*} of the outcome o^* and that it does not violate the condition of Definition 6 that there is no other possibly causing compound event for l^* occurring after ϵ_{i^*} and before σ_{j^*}. Therefore, ϵ_{i^*} is a causing step of l^* for transition state σ_{j^*} of θ in ρ whenever $pcce(i^*, l^*, j^*)$ is in A *and* $\neg cce(i^*, l^*, j^*)$ is not in A.

Finally, we consider once more the rules of group 9. The literal $directCause(e^*, i^*, l^*, j^*)$ is in A whenever $directEffect(l^*, e^*, i^*)$ and $cce(i^*, l^*, j^*)$ is in A. We also know that when $directEffect(l^*, e^*, i^*)$ is in A, then $l^* \in E(e^*, \sigma_{i^*})$. We also know that when $cce(i^*, l^*, j^*)$ is in A, then ϵ_{i^*} is a causing compound event of l^* for the transition state σ_{j^*} for outcome θ. In this case, the conditions of Definition 7 are satisfied and therefore the elementary event $e^* \in \epsilon_i^*$ is a direct cause of l^* for the transition state σ_{j^*} of θ in ρ when the literal $directCause(e^*, i^*, l^*, j^*)$ is in A. □

Next we, state and prove soundness and completeness of $\Pi_{indirect}$.

Theorem 2. *Let $\psi = \langle \theta, \rho, AD \rangle$ be a problem, ϵ_i be a compound event in ρ, σ_j be a transition state of θ in ρ, l be a literal in θ, and Π_ψ be a problem translation of ψ. Given the program $\Pi_{indirect} = \Pi_\psi \cup \Pi_{\mathcal{AL}} \cup \Pi_T \cup \Pi_C \cup \Pi_I$, compound event $\epsilon \in \epsilon_i$ is a indirect cause of l holding in σ_j if-and-only-if the atoms $indirectCause(c, i, l, j)$ and $subset(c, i)$ are in the answer set of $\Pi_{indirect}$, as well as an atom $inSubset(e, c)$ for every $e \in \epsilon$.*

Proof. Left-to-right.

We begin by characterizing an answer set A of $\Pi_{indirect}$ containing an atom $indirectCause(c, i, l, j)$: The answer set first contains the problem translation set $\alpha(\psi)$.

- Next, A contains an atom of form $transitionState(theta, j^*)$ for every transition state σ_{j^*} of θ in ρ,
- and an atom $transitionState(olit(l^*), i^*)$ for every transition state σ_i^* of the singleton set $\{l^*\}$ for $l^* \in \theta$.
- A also contains an atom $\neg ocSat(theta, i^*)$ for each state σ_{i^*} in ρ such that $\theta \not\subseteq \sigma_i^*$.
- Similarly, for $l^* \in \theta$ where $\{l^*\} \not\subseteq \sigma_{t^*}$ in ρ, A contains an atom $\neg ocSat(olit(l^*), t^*)$.
- Given a literal $l^* \in \theta$ and a transition state σ_{j^*} of θ in ρ, A contains an atom $pcce(i^*, l^*, j^*)$ for every possibly causing compound event ϵ_{i^*} of $\{l^*\}$ for σ_{j^*}.
- Given possibly causing compound events ϵ_{i^*} and $\epsilon_{i'^*}$ of $\{l^*\}$ for σ_{j^*}, A contains an atom $\neg cce(i^*, l^*, j^*)$ when $i^* < i'^*$.
- A also contains an atom $cce(i^*, l^*, j^*)$ for every causing compound event ϵ_i^* of a literal $l^* \in \theta$ for a transition state σ_j^* of θ.
- For every literal $l^* \in E(e^*, i^*)$ where elementary event e^* occurs at i^* in ρ, A also contains an atom $directEffect(l^*, e^*, i^*)$.
- A contains an atom $preservedPrec(s^*, l'^*, i'^*)$ for every state constraint s^* and $\sigma_{i'^*}$ such that $prec(s^*) \subseteq \sigma_{i'^*}$ and $l^* \in I(s^*, \sigma_{i^*})$.
- A contains an atom $directlyCausedPrec(s^*, \lambda(k^*, i^*), l'^*, i'^*)$ for every state constraint s^*, state $\sigma_{i'^*}$, and compound event $\epsilon_{k^*} \subseteq \epsilon_{i^*}$ such that $prec(s^*) \subseteq \sigma_{i'^*}$ and $l'^* \in M(s^*, \epsilon_{k^*}, \sigma_{i'^*})$.
- A also contains an atom $chain(\lambda(k^*, i^*), l^*, i'^*)$ for every static chain $\chi(\epsilon_{k^*}, l^*, \sigma_{i'^*})$.
- an atom $gamma(n^*, nil, \lambda(k^*, i^*), l^*, i'^*)$ for the static chain $\chi(\epsilon_{k^*}, l^*, \sigma_{i'^*})$
- For every link γ_g^* in a static chain $\chi(\epsilon_{k^*}, l^*, \sigma_{i'^*})$, A contains the following:
  - an atom $gamma(g^*, s^*, \lambda(k^*, i^*), l^*, i'^*)$ for every state constraint s^* in $\gamma_{j'^*}$
  - an atom $consquenceOfGamma(l'^*, g^*, \lambda(k^*, i^*), l^*, i'^*)$ for every $l'^* \in C(\gamma_{g^*})$

  - an atom $preconditionOfGamma(p'^*, g^*, \lambda(k^*, i^*), l^*, i'^*)$ for every $l'^* \in P(\gamma_{g^*})$
  - an atom $causedByGamma(s^*, l'^*, g^*, \lambda(k^*, i^*), l^*, i'^*)$ for every state constraint $s^* \in \gamma_{g^*}$ and $l'^* \in prec(s^*)$

- Given a chain $\chi(\epsilon_{k^*}, l^*, \sigma_{i'^*}) = \langle \gamma_1, \ldots, \gamma_{n'^*} \rangle$, A contains the atom $contributed(e^*, p'^*, \lambda(k^*, i^*), l^*, i'^*)$ when $p'^* \in P(\{\gamma_1, \ldots, \gamma_{n^*}\})$, $e^* \in \epsilon_i^*$, and $p' \in E(e^*, \sigma_{i^*-1})$.

- Given a chain $\chi(\epsilon_{k^*}, l^*, \sigma_{i'^*}) = \langle \gamma_1, \ldots, \gamma_{n^*} \rangle$, A contains an atom $extraEvents(\lambda(k^*, i^*), l^*, \sigma_{i'^*})$ if there is some $e^* \in \epsilon_{k^*}$ such that $E(e^*, \sigma_{i^*}) \cap P(\{\gamma_1, \ldots, \gamma_{n^*}\}) = \emptyset$.

- Finally, the set A contains an atom $indirectCause(\lambda(k^*, i^*), i^*, l^*, j^*)$ for every compound event $\epsilon_{k^*} \subseteq \epsilon_{i^*}$ that is an indirect cause of l^* for transition state σ_{j^*} of θ.

Let us show that if ϵ' is an indirect cause of l holding in σ_j, then A is an answer set of $\Pi_{indirect}$ by proving that A is the minimal set of atoms closed under the rules of the reduct $\Pi^A_{indirect}$. As in the proof of Theorem 1, we give a simplified form of the reduct where the occurrences of atoms of the form $next(i, j)$ have been unfolded into expressions $j = i + 1$. $\Pi^A_{indirect}$ contains:

1. set $\alpha(\psi)$
2. all rules in $\Pi_{\mathcal{AL}}$
3. rules of type (3) through (8) in the reduct Π^A_{direct} from Theorem 1.
4. a rule of the form

$$\begin{aligned} preservedPrec(s^*, l'^*, i'^*) \leftarrow \quad & s\_law(s^*), prec\_h(s^*, i'^*), \\ & prec(s^*, x^*, l'^*), holds(l'^*, i^*), \\ & i'^* = i^* + 1. \end{aligned}$$

   for all state constraints s^*, fluent literals l^*, and integers i^*, i'^*, and x^*.

5. a rule of the form

$$\begin{aligned} directlyCausedPrec(s^*, c^*, l^*, i'^*) \leftarrow \quad & s\_law(s^*), prec\_h(s^*, i^*), inSubset(e^*, c^*), \\ & prec(s^*, x^*, l'^*), directEffect(l^*, e^*, i^*), \\ & holds(\overline{l^*}, i^*), i'^* = i^* + 1. \end{aligned}$$

for all state constraints constraints s^*, fluent literals l^*, elementary events e^*, subsets c^*, and integers i^*, i'^*, and x^*.

6. a rule of the form

$$\begin{aligned} gamma(1, s^*, c^*, l^*, i'^*) \leftarrow \quad & subset(c^*, i^*), holds(l^*, i'^*), s\_law(s^*), \\ & prec\_h(s^*, i'^*), i'^* = i^* + 1, \\ & \#count\{l_1^* : preservedPrec(s^*, l_1^*, i'^*)\} = n_1^*, \\ & \#count\{l_2^* : directlyCausedPrec(s^*, c^*, l_2^*, i'^*)\} = n_2^*, \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad n_2^* > 0, \\ & \#count\{l_3^* : prec(s^*, x^*, l_3^*), l_3^*! = nil\} = n_3^*, \\ & n_3^* = n_1^* + n_2^*. \end{aligned}$$

for all state constraints constraints s^*, fluent literals l^*, elementary events e^*, subsets c^*, and integers i^*, i'^*,g^* l_1^*, l_2^*, l_3^*, n_1^*, n_2^*, n_3^*, and x^*.

7. a rule of the form

$$\begin{aligned} gamma(g^*+1, s^*, c^*, l^*, i'^*) \leftarrow & subset(c^*, i^*), holds(l^*, i'^*), s\_law(s^*), \\ & prec\_h(s^*, i'^*), i'^* = i^* + 1, \\ & gamma(g^*, x^*, c^*, l^*, i'^*), \\ & \#count\{l_1^* : preservedPrec(s, l_1^*, i'^*)\} = n_1^*, \\ & \#count\{l_2^* : directlyCausedPrec(s^*, c^*, l_2^*, i'^*)\} = n_2^*, \\ & \#count\{l_3^* : causedByGamma(s^*, l_3^*, g^*, c^*, l^*, i'^*)\} = n_3^*, \\ & n_3^* > 0, \\ & \#count\{l_4^* : causedByGamma(s^*, l_4^*, g'^*, c^*, l^*, i'^*), \\ & \qquad g'^* <= (g^* - 1)\} = n_4^*, \\ & \#count\{l_5^* : prec(s^*, x^*, l_5^*, l_5^* \,! = nil\} = n_5^*, \\ & n_5^* = n_1^* + n_2^* + n_3^* + n_4^*, \end{aligned}$$

for all state constraints constraints s^*, fluent literals l^*, elementary events e^*, subsets c^*, and integers i^*, i'^*, l_1^*, l_2^*, l_3^*, l_4^*, l_5^*, n_1^*, n_2^*, n_3^*, n_4^*, n_5^*, g^*, and x^* such that $consequenceOfGamma(l^*, g^*, c^*, l^*, i^*)$ is not in A.

8. a rule of the form

$$consequenceOfGamma(l'^*, g^*, c^*, l^*, i'^*) \leftarrow \quad gamma(g^*, s^*, c^*, l^*, i'^*), head(s^*, l'^*).$$

for all state constraints s^*, fluent literals l^* and l'^*, subsets c^*, and integers i^* and g^*.

9. a rule of the form

$$\begin{aligned} preconditionOfGamma(p^*, g^*, c^*, l^*, i'^*) \leftarrow \quad & gamma(g^*, s^*, c^*, l^*, i'^*), \\ & prec(s^*, x^*, p^*), p^*! = nil. \end{aligned}$$

for all state constraints s^*, fluent literals l^* and p^*, subsets c^*, and integers i^*, g^*, and x^*.

10. a rule of the form

$$\begin{aligned} causedByGamma(s^*, l'^*, g^*, c^*, l^*, i'^*) \leftarrow \quad & s\_law(s^*), prec\_h(s^*, i'^*), \\ & prec(s^*, x^*, l'^*), \\ & consequenceOfGamma(l'^*, g^*, c^*, l^*, i'^*). \end{aligned}$$

for all state constraints s^*, fluent literals l^* and l'^*, subsets c^*, and integers i^*, g^*, and x^*.

11. a rule of the form

$$\begin{aligned} chain(c^*, l^*, i'^*) \leftarrow \quad & gamma(n^*, s^*, c^*, l^*, i'^*), \\ & consequenceOfGamma(l^*, n^*, c^*, l^*, i'^*). \end{aligned}$$

for all state constraints s^*, fluent literals l^*, subsets c^*, and integers i^* and n^*.

12. a rule of the form

$$\begin{aligned} contributed(e^*, p^*, c^*, l^*, i'^*) \leftarrow \quad & preconditionOfGamma(p^*, y^*, c^*, l^*, i'^*), \\ & inSubset(e^*, c^*), directEffect(p^*, e^*, i^*), \\ & i'^* = i^* + 1. \end{aligned}$$

for all fluent literals l^* and p^*, subsets c^*, elementary events e^*, and integers i^*, i'^* and y^*.

13. a rule of the form

$$\begin{aligned} extraEventsInSubset(c^*, l^*, i^*) \leftarrow \quad & chain(c^*, l^*, i'^*), \\ & inSubset(e^*, c^*), \\ & i'^* = i^* + 1. \end{aligned} \tag{5.43}$$

for all fluent literals l^*, subsets c^*, elementary events e^*, and integers i^*, i'^* and j^* such that $contributed(e^*, p^*, c^*, l^*, i'^*) \notin A$.

14. a rule of the form

$$indirectCause(c^*, i^*, l^*, j^*) \leftarrow \quad cce(i^*, l^*, j^*),$$
$$chain(c^*, l^*, i'^*), i'^* = i^* + 1.$$

for all fluent literals l^*, subsets c^*, and integers i^*, i'^*, and j^* such that $extraEventsInSubset(c^*, l^*, i'^*)$ is not in A.

A is closed under $\Pi^A_{indirect}$. We will prove it for every rule of the program.

Rules of group [1]: obvious.

Rules of group [2)]: The conclusion follows from Theorem 1 from [43].

Rules of group [3)]: The conclusion follows from Theorem 1 from this dissertation.

Rules of group [4]: If the body of the rule is in A, then there exists a state constraint $s^* : \ l_0^*$ **if** ω^* in AD such that $\omega^* \subseteq \sigma_{i'^*}$ and a literal $l'^* \in \omega^*$ such that $l'^* \in \sigma_{i^*}$ where $i'^* = i^* + 1$. It is straightforward to verify that l'^* is also in the set $I(s^*, \sigma_{i'^*})$. Therefore, the atom $preservedPrec(s^*, l'^*, i'^*) \in A$ by construction of A.

Rules of group [5]: If the body of the rule is in A, then there exists a state constraint $s^* : \ l_0^*$ **if** ω^* in AD such that $\omega^* \subseteq \sigma_{i'^*}$, a literal $l'^* \in \omega^*$ such that $\overline{l^*} \in \sigma_{i^*}$ where $i'^* = i^* + 1$, and a subset e_{k^*} corresponding to the subset identifier c^* such that $l'^* \in E(e_{k^*}, \sigma_{i^*})$. It is straightforward to verify that $l'^* \in D(s^*, \sigma_{i^*})$. By construction of A, the atom $directlyCausedPrec(s^*, \lambda(k^*, i^*), l'^*, i'^*)$ is in A.

Rules of group [6]: If the body of the rule is in A, then there exists a state constraint $s^* : l'^*$ **if** ω^* such that $prec(s^*) \in \sigma_{i'^*}$, a subset $\lambda(k^*, i^*)$ corresponding to identifier c^*, and $i'^* = i^* + 1$. If line 3 of this rule is satisfied, then the cardinality of the set of atoms in A of form $preservedPrec(s^*, l_1^*, i'^*)$ is n_1^*. Similarly, if line 4 is satisfied, then the cardinality of the set of atoms in A of form $directlyCaused-Prec(s^*, c^*, l_1^*, i'^*)$ is n_2^*, where $n_2^* > 0$, and if line 6 is satisfied then the cardinality of the set of atoms in A of form $prec(s^*, x^*, l_3^*)$ is n_3^*. Finally, if line 7 is satisfied, then $n_3^* = n_1^* + n_2^*$. Recall that $l'^* \in I(s^*, \sigma_{i'^*})$ when $preservedPrec(s^*, l'^*, i'^*)$ and a literal $l'^* \in M(s^*, c^*, i'^*)$ when $directlyCausedPrec(s^*, c^*, l'^*, i'^*)$ in A. We conclude that Condition 1 of Definition 9 is satisfied and there is a link γ_1 in the static chain $\chi(c^*, l^*, i'^*)$ which contains a state constraint s^* for every atom in A of the form $gamma(1, s^*, c^*, l^*, i'^*)$.

Rules of groups [7], (8), and (10): Leveraging what we concluded about rule of group (6), one can check by induction that A is closed under the rules of these groups.

Rules of group [11]: If the body is satisfied, it means that there is some γ_{n^*} in the static chain $\chi(c^*, l^*, i'^*)$ and l'^* belongs to the set $C(\gamma_{n^*})$. By Definition 9, we have shown that this is a static chain, and therefore the head is in A by $\alpha(\psi)$.

Rules of group [12]: It the body of this rule is satisfied then there is a subset an elementary event $e^* \subseteq \epsilon_{k^*})$ (via subset identifier c^*) such that $p^* \in E(e^*, \sigma_{i^*})$. Leveraging reasoning similar to that used for proving closure for rules of group [8], we see that for rules of group [9], if $preconditionOf-Gamma(p^*, g^*, c^*, l^*, i'^*)$ is in A, then $p^* \in P(\gamma_{g^*})$ in the static chain $\chi(c^*, l^*, i^*)$ where $i'^* = i^* + 1$. If line 1 of this rule is satisfied, then e^* directly caused $p^* \in P(\{\gamma_1, \ldots, \gamma_{y^*}, \ldots, \gamma_{n^*}\})$ to hold. By construction of A, $contributed(e^*, p'^*, c^*, l^*, i'^*) \in A$.

Rules of group [13]: If the rule is in A, then $contributed(e^*, p'^*, \lambda(k^*, i^*), l^*, i'^*)$ is not in A. If the body of the rule is satisfied, conclusions from earlier considered groups can be leveraged

to conclude that there is an elementary event $e^* \in \epsilon_{k^*}$, where $\epsilon_{k^*} \subseteq \epsilon_{k^*}$, such that $E(e^*, \sigma_{i^*}) \cap P(\{\gamma_1, \ldots, \gamma_{n^*}\}) = \emptyset$. By construction of A, the atom $extraEvents(\lambda(k^*, i^*), l^*, \sigma_{i'^*})$ is in A.

Rules of group [14]: If the rule is in A, then $extraEvents(\lambda(k^*, i^*), l^*, \sigma_{i'^*})$ is not in A, satisfying condition 2 of Definition 10. If the body is satsified, then $\epsilon_{k^*} \subseteq \epsilon_{i^*}$ is a causing compound event of l for the transition state σ_{j^*} of θ in ρ, $\chi(c^*, l^*, i^*)$ is a static chain, and $i'^* = i^* + 1$. We conclude that Condition 1 of Definition 10 is satisfied, hence $\epsilon_{k^*} \subseteq \epsilon_{i^*}$ in ρ is an indirect cause of l^* for transition state θ in ρ. The elementary events in ϵ_{k^*} corresponds to the set of all e^* such that an atom $inSubset(e^*, c^*)$ is in A.

A is the minimal set closed under the rules of $\Pi^A_{indirect}$. We will prove this by assuming that there exists a set $B \subseteq A$ such that B is closed under the rules of $\Pi^A_{indirect}$, and by showing that $B = A$ using an approach similar to that used in proving Theorem 1.
First, the set $\alpha(\psi) \subset B$ since these are the facts of $\Pi^A_{indirect}$.

Rules of group [2]: It is possible to apply the Splitting Set Theorem [56, 57] to Π_{direct} so that the bottom of the program corresponds to $\alpha(SD, \Gamma_D)$ from [43]. The restriction of A to the signature of $\alpha(SD, \Gamma_D)$ is an answer set of $\alpha(SD, \Gamma_D)$ by Theorem 1 from [43] and therefore is minimal. This tells us that whenever the literals of the form $holds(\cdot, \cdot)$, $occurs(\cdot, \cdot)$, $step(\cdot)$, $next(\cdot, \cdot)$, $prec\_h(\cdot, \cdot)$, and $all\_h(\cdot, \cdot, \cdot)$ are in A, then they must also be in B.

Rules of group [3]: Using reasoning analogous to the proof of Theorem 1 of this dissertation, we can prove that all atoms in A are also in B for rules of group [3].

Rules of group [4]: If the head of the rule is in A, then by supportedness the body is also satisfied in A. By Theorem 1 of [43] and because the atoms of A and B coincide, the body is also in the set B. By closedness of B, the head is also in B.

Rules of group [5]: When the head is in A, then again by supportedness the body must also be satisfied in A. By Theorem 1 of [43], the facts of $\alpha(\psi)$, and leveraging the observation that the heads of rules in group [3] coincide in A and B, then the body of rules of this group must also be in B. Because B is closed under the reduct, the head must also be in B.

Rules of group [6]: If the head of a rule of this group is in the set A, then by supportedness the body must also be satisfied in A. Using our conclusions about coincidence of heads of rules of groups [4] and [5], Theorem 1 of [43], and the coincidence of facts in A and B, then the body must also be in B. By closedness of B, the head must also be in B.

Rules of groups [7], [8], [10[]: Leveraging our conclusions about rules of group [6], the facts coinciding in A and B, and Theorem 1 of [43], it can be verified by induction that the heads of these rules coincide in groups A and B.

Rules of group [11]: If the head is in A, then the body is satisfied A by supportedness. Using our conclusions from reasoning about rules of groups rules of groups [6], [7] and [8], the body of the rule must also be satisfied in B, and the head is in B because it is closed under the reduct.

Rules of group [9]: When the head is in A, then the body must be satisfied in A by supportedness. Because the facts in $\alpha(\psi)$ and heads of rules of groups [6] and [7] coincide in sets A and B, then the body is satisfied in B. By closedness, the head is also in B.

Rules of group [12]: If the head is in A, then, again by supportedness, the body must be satisfied in A. Because the heads of rules in groups [1], [3], and [9] coincide in sets A and B, then the body must also be satisfied in B. The head is also in B because B is closed under the reduct.

Rules of group [13]: If the head of the rule is in A, then the body is satisfied in A by supportedness. Because the heads of rules of groups [1], [6], and [7] coincide in A and B, the body is also satisfied in B. Therefore, by closedness, head must also be in B.

Rules of group [14]: If the head of the rule is in A, then the body must also be satisfied in A. We know that when $cce(i^*, l^*, j^*)$ is in A it is also in B because of our consideration of rules of group [3], and we have already concluded that the heads of rules of group [11] coincide in A and B. Therefore, the body is also in B, and by closedness the head is also in B.

We have just proven that for every rule r of the reduct, if $head(r)$ belongs to A, then $head(r)$ belongs to B. By the supportedness property, every literal $l \in A$ must be in the head of some rule. Hence, all literals of A also belong to B. Therefore, $A = B$. We have proven that A is the minimal set of atoms closed under the rules of the reduct $\Pi^A_{indirect}$.

Right-to-left.

Let A be an answer set of $\Pi_{indirect}$ and A include atoms $indirectCause(c, i, l, j)$, $subset(c, i)$, as well as atoms for all elementary events e^* such that $inSubset(e^*, c) \in A$. We will show then $\epsilon' \in \epsilon_i$ is an indirect cause for the literal l holding in the transition state σ_j of θ from ψ, where ϵ' is the set of all e^*.

Because A is an answer set of $\Pi_{indirect}$, A is also an answer set of the reduct $\Pi^A_{indirect}$. We will consider the groups [1] through [14] as the reduct.

Rules of group [1]: obvious.

Rules of group [2]: The conclusion follows from Theorem 1 from [43].

Rules of group [3]: The conclusion follows from Theorem 1 from this dissertation.

Consider first rules of group [14]. If the atom $indirectCause(c, i, l, j)$ is in A, then by construction of the reduct $\Pi^A_{indirect}$, then $extraEventsInSubset(c^*, l^*, i^*)$ must not be in A. Additionally, the body of the corresponding rule in group [14] must be satisfied in A by the supportedness property because A is closed under $\Pi^A_{indirect}$. Therefore, the atoms $cce(i^*, l^*, j^*)$ and $chain(c^*, l^*, i'^*)$ are in A such that $i'^* = i^* + 1$. Using reasoning analogous to the proof of Theorem 1, i^* must be a causing compound event of l^* for σ_{j^*} because $cce(i^*, l^*, j^*) \in A$. Also, by construction of the reduct, the atom $extraEventsInSubset(c^*, l^*, i'^*)$ is not in A. Similarly for rules of group [11], we have already determined that atom $chain(c^*, l^*, i'^*) \in A$, therefore the atoms $gamma(1, s^*, c^*, l^*, i'^*)$ and $consequenceOfGamma(l^*, n^*, c^*, l^*, i'^*)$ are also in A by supportedness.

Consider now rules of group [6]. We have concluded that the atom $gamma(1, s^*, c^*, l^*, i'^*)$ is in A. By supportedness, A must include an atom $subset(c^*, i^*)$. There must be a compound event $\epsilon_{k^*} \subseteq \epsilon_{i^*}$ in the path ρ by construction of the problem translation $\alpha(\psi)$. By the semantics of $\mathcal{AL}$, there must also be states σ_{i^*} and $\sigma_{i'^*}$ in ρ. A also includes atom $holds(l^*, i'^*)$ and, by $\alpha(\psi)$, $l^* \in \sigma_{i'^*}$.

Now let us consider the $\#count$ aggregates from group [6]. Their atoms are obtained from rules of group [4] and rules of group [5]. For the first group, if $preservedPrecs(s^*, l'^*, i'^*) \in A$, then by supportedness and from Theorem 1 of [43], we can conclude that there is a state constraint s^* whose preconditions are satisfied in the state $\sigma_{i'^*}$, $l'^* \in prec(s^*)$ and therefore $l'^* \in \sigma_{i'^*}$. From the construction of $\alpha(\psi)$ we can further reason that $l'^* \in \sigma_{i^*}$. Because we know that $l'^* \in \sigma_{i'^*}$, $l'^* \in \sigma_{i^*}$, and $l'^* \in prec(s^*)$ is also true, we can conclude that l'^* is in the set of s^*'s preserved preconditions $I(s, \sigma_{i'^*})$ if $preservedPrecs(s^*, l'^*, i'^*) \in A$.

For group [5], our earlier conclusions about $\sigma_{i'^*}$ and ϵ_{k^*} can be leveraged together with reasoning analogous to the proof of Theorem 1 of this dissertation, Theorem 1 from [43], as well as the translation $\alpha(\psi)$ to conclude that if an atom $directlyCausedPrec(s^*, e^*, l'^*, i^*)$ belongs to A, then

$l'^* \in prec(s^*)$ and $l'^* \in E(e^*, \sigma_{i^*})$ for every elementary event $e^* \subseteq \epsilon_{k^*}$. It can be further reasoned using these conclusions and the earlier conclusion that $l^* \notin \sigma_{i^*}$ that l'^* is a member of the set of s^*'s directly caused preconditions in σ_{i^*} $M(s^*, \epsilon_{k^*}, \sigma_{i'^*})$ when the atom $directlyCausedPrec(s^*, e^*, l'^*, i^*) \in A$.

Returning to rules of group [6], we find that A must also contain atoms $s\_law(s^*)$, one ore more atoms of the form $prec(s^*, x^*, l'^*)$ as well as an atom $prec\_h(s^*, i'^*)$. By Theorem 1 of [43], we can conclude that there is a state constraint s^* whose preconditions are satisfied in the state $\sigma_{i'^*}$. Next, recall that by the semantics of the aggregate operator $\#count$, an aggregate literal is satisfied if the cardinality of its argument with respect to A is equal to the term u of the aggregate atom. Because we know that the body of this rule is satisfied by supportedness from our earlier consideration of rules of group [11], it must be the case that A contains n_3^* atoms of the form $prec(s^*, x^*, l^*)$ such that $l^* \neq nil$. Additionally, $n_2^* \geq 1$ and therefore there is at least one atom of the form $directlyCausedPrec(s^*, e^*, l'^*, i'^*)$ in A. We can now reason that if the body of a rule in this group is satisfied in A, then n_1^* is the cardinality of $I(s, \sigma_{i'^*})$ from our consideration of rules of group 4, n_2^* is the cardinality of $M(s^*, \epsilon_{k^*}, \sigma_{i'^*})$ from our consideration of rules of group [5], and n_3^* is the cardinality of atoms $prec(s^*, x^*, l'^*) \in A$ by $\alpha(\psi)$. From here, we conclude that Condition 1 of Definition 9 for the chain $\chi(c^*, l^*, i'^*)$ is satisfied because there is no overlap among the preconditions of s^* that have been preserved by inertia and those that have been caused directly. We have shown that if $gamma(1, s^*, c^*, l^*, i'^*) \in A$, there exists a sequence $\langle \gamma_1, \cdots \rangle$ satisfying Condition 1 of Definition 9 and $s^* \in \gamma_1$.

For rules of group [7], we it can be checked by induction and reasoning analogous to rules of group [6] that when $gamma(g^*, s'^*, c^*, l^*, i'^*) \in A$, there exists a sequence $\langle \gamma_1, \ldots, \gamma_{g^*}, \cdots \rangle$ such that Condition 2b of Definition 9 is satisfied for the no overlap case[1] and that $s'^* \in \gamma_{g^*}$. Moreover, we conclude that when $consequenceOfGamma(l'^*, g^*, c^*, l^*, i'^*)$ and $causedByGamma(s^*, l'^*, g^*, l^*, i'^*)$

[1]Recall that testing for such cases is trivial and adding the support to the program does not significantly change the proof strategy for a similar theorem.

are in A, $l'^* \in P(\gamma_{g^*})$, where $l^* \in prec(s^*)$.

Considering now rules of group [11], recall that earlier we concluded that $chain(c^*, l^*, i^*)$ is in A. Therefore, atoms $gamma(n^*, s^*, c^*, l^*, i'^*)$ and $consequenceOfGamma(l^*, n^*, c^*, l^*, i^*)$ are also in A. As we have concluded above, we know that because these atoms are A, there exists a sequence $\langle \gamma_1, \ldots, \gamma_{g^*}, \cdots \rangle$ such that $l^* \in C(\gamma_{g^*})$ (i.e., l^* is a consequence of γ_{g^*}), making it the final link of the chain by Definition 9. Therefore, we can conclude that when $chain(c^*, l^*, i^*)$ is in A, there exists a static chain $\chi(c^*, l^*, i^*) = \langle \gamma_1, \ldots, \gamma_{n^*} \rangle$.

Before returning to rules of group [14], we draw attention to the fact that it is straightforward to demonstrate by considering rules of groups [9], [12], and [13] that if the atom $extraEventsInSubset$ (c^*, l^*, i^*) is in A, then there exists at least one elementary event $e^* \in e_{k^*}$ such that $E(e^*, \sigma_{i^*}) \cap P(\{\gamma_1, \ldots, \gamma_{n^*}\}) \neq \emptyset$. The verification is achieved using similar reasoning as used for earlier considered groups, Theorem 1 from [43], the construction of the reduct, and the supportedness property.

Once more considering rules of group [14], recall that that the atoms $cce(i^*, l^*, j^*)$ and $chain(c^*,$ $l^*, i^*)$ are in A by supportedness, and that ϵ_i must be a causing compound event of l^* for the transition state σ_j of the outcome θ, from reasoning analogous to the proof of Theorem 1 of this dissertation. From our earlier conclusion, whenever $chain(c^*, l^*, i^*) \in A$, there exists a static chain $\chi(c^*, l^*, i^*) = \langle \gamma_1, \ldots, \gamma_n^* \rangle$ which satisfies Condition 1 of the definition of indirect cause for compound event $e_k \in \epsilon_i$, literal l^*, and the transition state σ_{j^*} of θ. By construction of the reduct, the atom $extraEventsIn(c^*, i^*, l^*)$ is not in A, therefore Condition 2 of Definition 10 is satisfied, alowing us to conclude that when the atom $indirectCause(c^*, i^*, l^*, j^*)$ is in A, then the compound event e_{k^*} corresponding to c^* is an indirect cause of l^* for the transition state σ_{j^*} of θ in ρ. □

We have guaranteed the correctness of the implementation for direct cause and the improved version of indirect cause. Next, we present two examples from Chapter 4 encoded in ASP.

5.3 ASP Examples

5.3.1 Extended Yale Shooting Problem

Next, we present the translation of the problem ψ_Y from Chapter 4 translated to ASP without the translation of ψ_Y's outcome $\theta_Y = \{\neg isAlive(turkey)\}$. We then give the ASP translation θ_Y, θ'_Y, and θ''_Y, and show that the answer set of $\Pi_{direct} \cup \alpha(\psi)$ unioned with each outcome contains the expected cause.

YSP Problem Instance $\alpha(\psi_Y)$:

```
%%%%%% EVENTS %%%%%%

event(handsGun(tommy)).

event(loads(suzy,gun)).

event(shoots(suzy,turkey)).


occurs(handsGunTo(suzy),1).

occurs(loads(suzy,gun),2).

occurs(shoots(suzy,turkey),3).


step(1..4).


subset(c1,1).

inSubset(handsGunTo(suzy),c1).


subset(c2,2).

inSubset(loads(suzy,gun),c2).


subset(c3,3).

inSubset(shoots(suzy,turkey),c3).
```

```
%%%%%% STATES %%%%%%

holds(isAlive(turkey),1).

holds(neg(isLoaded(gun)),1).

holds(neg(hasGun(suzy)),1).


holds(isAlive(turkey),2).

holds(isLoaded(gun),2).

holds(neg(hasGun(suzy)),2).


holds(isAlive(turkey),3).

holds(isLoaded(gun),3).

holds(hasGun(suzy),3).


fluent(isAlive(turkey)).

fluent(hasGun(suzy)).

fluent(isLoaded(gun)).


%%%%%% ACTION DESCRIPTION %%%%%%
```

```
d_law(d1).

head(d1,hasGun(suzy)).

event(d1,handsGun(tommy)).

prec(d1,1,nil).

d_law(d2).

head(d2,isLoaded(gun)).

event(d2,loads(suzy,gun)).

prec(d2,1,neg(isLoaded(gun))).

prec(d2,2,nil).

d_law(d3).

head(d3,neg(isAlive(turkey))).

event(d3,shoots(suzy,turkey)).

prec(d3,1,isAlive(turkey)).

prec(d3,2,nil).

d_law(d4).

head(d4,neg(isAlive(turkey))).

event(d4,shoots(suzy,turkey)).

prec(d4,1,isAlive(turkey)).

prec(d4,2,nil).

i_law(im1).

head(im1,neg(isAlive(turkey))).

prec(im1,1,neg(isLoaded(gun))).

prec(im1,2,nil).
```

The translation of $\alpha(\theta_Y)$ of $\theta_Y = \{\neg isAlive(turkey)\}$ is:

```
outcome(theta).

inOutcome(neg(isAlive(turkey)),theta).

outcome(olit(neg(isAlive(turkey)))).

inOutcome(neg(isAlive(turkey)),olit(neg(isAlive(turkey)))).
```

and the answer set of $\Pi_{direct} \cup \alpha(\psi_Y) \cup \alpha(\theta_Y)$ contains:

```
directcause(shoots(suzy,turkey),3,neg(isAlive(turkey)),4).
```

Theorem 1 guarantees that $shoots(suzy, turkey) \in \epsilon_3$ is a direct cause of $\neg isAlive(turkey)$ for state σ_4. The translation of $\alpha(\theta'_Y)$ of $\theta'_Y = \{isLoaded(gun)\}$ is:

```
outcome(theta).

inOutcome(neg(isAlive(turkey)),theta).

outcome(olit(neg(isAlive(turkey)))).
```

```
inOutcome(neg(isAlive(turkey)),olit(neg(isAlive(turkey)))).
```

and the answer set of $\Pi_{direct} \cup \alpha(\psi_Y) \cup \alpha(\theta'_Y)$ contains:

```
directcause(loads(suzy,gun),2,isLoaded(gun),3).
```

Theorem 1 guarantees that $loads(suzy, gun) \in \epsilon_2$ is a direct cause of $isLoaded(gun)$ for state σ_3. The translation of $\alpha(\theta''_Y)$ of $\theta''_Y = \{hasGun(suzy)\}$ is:

```
outcome(theta).
inOutcome(hasGun(suzy),theta).
outcome(olit(hasGun(suzy))).
inOutcome(hasGun(suzy),olit(hasGun(suzy))).
```

and the answer set of $\Pi_{direct} \cup \alpha(\psi_Y) \cup \alpha(\theta''_Y)$ contains:

```
directcause(handsGun(tommy),1,hasGun(suzy),2).
```

Theorem 1 guarantees that $handsGun(tommy) \in \epsilon_1$ is a direct cause of $hasGun(suzy)$ for state σ_3.

5.3.2 Self-driving Car Problem

Next, we present the translation of the problem ψ_S from Chapter 4 translated to ASP without the translation of ψ_S's outcome $\theta_S = \{\neg error(crash)\}$. We then give the ASP translation θ_S, θ'_S, and θ''_S. Finally, we show that the answer set of $\Pi_{indirect} \cup \alpha(\psi)$ unioned with $\alpha(theta_S)$ contains the expected cause, and similarly that $\Pi_{direct} \cup \alpha(\psi)$ unioned with the translations of $\alpha(\theta'_S)$ and $\alpha(\theta''_S)$ contain the expected causes.

Self-driving Car Problem Instance $\alpha(\psi_S)$:

```
%%%%%% EVENTS %%%%%%
occurs(report(obst,o1),1).
occurs(report(analyze,d1),1).
occurs(report(d(accel),d1),2).
occurs(report(analyze,o1),2).
occurs(report(obst,o2),3).
occurs(report(analyze,d2),3).
occurs(report(act(accel),c1),3).
```

```
event(report(obst,o1)).
event(report(analyze,d1)).
event(report(d(accel),d1)).
event(report(analyze,o1)).
event(report(obst,o2)).
event(report(analyze,d2)).
event(report(act(accel),c1)).
step(1..4).

subset(c1,1).
inSubset(report(obst,o1),c1).

subset(c2,1).
inSubset(report(analyze,d1),c2).

subset(c3,1).
inSubset(report(obst,o1),c3).
inSubset(report(analyze,d1),c3).

subset(c4,2).
inSubset(report(analyze,o1),c4).

subset(c5,2).
inSubset(report(d(accel),d1),c5).

subset(c6,2).
inSubset(report(analyze,o1),c6).
inSubset(report(d(accel),d1),c6).

subset(c7,3).
inSubset(report(obst,o2),c7).

subset(c8,3).
inSubset(report(act(accel),c1),c8).

subset(c9,3).
inSubset(report(analyze,d2),c9).

subset(c10,3).
inSubset(report(obst,o2),c10).
inSubset(report(act(accel),c1),c10).

subset(c11,3).
inSubset(report(obst,o2),c11).
inSubset(report(analyze,d2),c11).

subset(c12,3).
inSubset(report(act(accel),c1),c12).
inSubset(report(analyze,d2),c12).

subset(c13,3).
inSubset(report(obst,o2),c13).
```

```
inSubset(report(act(accel),c1),c13).
inSubset(report(analyze,d2),c13).

%%%%%% STATES %%%%%%
holds(neg(error(crash)),1).
holds(neg(infer(crash)),1).
holds(neg(known(obst)),1).
holds(neg(mustAct(accel)),1).
holds(obst(o1),1).
holds(obst(o2),1).
holds(dec(d1),1).
holds(dec(d2),1).
holds(control(c1),1).

holds(neg(error(crash)),2).
holds(neg(infer(crash)),2).
holds(known(obst),2).
holds(neg(mustAct(accel)),2).
holds(obst(o1),2).
holds(obst(o2),2).
holds(dec(d1),2).
holds(dec(d2),2).
holds(control(c1),2).

holds(neg(error(crash)),3).
holds(neg(infer(crash)),3).
holds(known(obst),3).
holds(mustAct(accel),3).
holds(obst(o1),3).
holds(obst(o2),3).
holds(dec(d1),3).
holds(dec(d2),3).
holds(control(c1),3).

holds(error(crash),4).
holds(infer(crash),4).
holds(known(obst),4).
holds(mustAct(accel),4).
holds(obst(o1),4).
holds(obst(o2),4).
holds(dec(d1),4).
holds(dec(d2),4).
holds(control(c1),4).

fluent(dec(d2)).
fluent(infer(crash)).
fluent(obst(o1)).
fluent(obst(o2)).
fluent(error(crash)).
fluent(control(c1)).
fluent(known(obst)).
fluent(dec(d1)).
fluent(mustAct(accel)).
```

```
%%%%%% ACTION DESCRIPTION %%%%%%

d_law(d1).                              d_law(d3).

head(d1,known(obst)).                   head(d3,infer(crash)).

event(d1,report(obst,o1)).              event(d3,report(act(accel),c1)).

prec(d1,1,obst(o1)).                    prec(d3,1,known(obst)).

prec(d1,2,nil).                         prec(d3,2,nil).


d_law(d2).                              s_law(s1).

head(d2,mustAct(accel)).                head(s1,error(crash)).

event(d2,report(d(accel),d1)).          prec(s1,1,infer(crash)).

prec(d2,1,dec(d1)).                     prec(s1,2,nil).

prec(d2,2,nil).
```

The translation of $\alpha(\theta_S)$ of $\theta_Y = \{error(crash)\}$ is:

```
outcome(theta).

inOutcome(error(crash),theta).

outcome(olit(error(crash))).

inOutcome(error(crash),olit(error(crash))).
```

and the answer set of $\Pi_{indirect} \cup \alpha(\psi_Y) \cup \alpha(\theta_Y)$ contains:

```
indirectcause(c8,3,error(crash),4).

inSubset(report(act(accel),c1),c8).
```

Theorem 2 guarantees that $\{report(act(accel, c1)\} \subset \epsilon_3$ is an indirect cause of $error(crash)$ for state σ_4. The translation of $\alpha(\theta'_S)$ of $\theta'_S = \{mustAct(accel)\}$ is:

```
outcome(theta).

inOutcome(mustAct(accel),theta).

outcome(olit(mustAct(accel))).

inOutcome(mustAct(accel),olit(mustAct(accel))).
```

and the answer set of $\Pi_{direct} \cup \alpha(\psi_Y) \cup \alpha(\theta'_Y)$ contains:

```
directcause(report(d(accel),d1),2,mustAct(accel),3).
```

Theorem 1 guarantees that $report(d(accel), d1) \in \epsilon_2$ is an indirect cause of $mustAct(accel)$ for state σ_3. Finally, the translation of $\alpha(\theta''_S)$ of $\theta''_S = \{mustAct(accel)\}$ is:

```
outcome(theta).
inOutcome(known(obst),theta).
outcome(olit(known(obst))).
inOutcome(known(obst),olit(known(obst))).
```

and the answer set of $\Pi_{direct} \cup \alpha(\psi_Y) \cup \alpha(\theta''_Y)$ contains:

```
directcause(report(obst,o1),1,known(obst),2).
```

Theorem 1 guarantees that $report(obst, o1) \in \epsilon_1$ is an indirect cause of $known(obst)$ for state σ_2. For both of the above examples, the implementations return answers that are in line both with intuition and the results of the theoretical framework.

In this chapter, we presented an implementation of the theoretical framework, proved the soundness and completeness of the implementation for the definition of direct cause and the improved definition of indirect cause. We have also presented the ASP encodings of the extended Yale Shooting Problem and the self-driving car example from Chapter 4. We next explore the implementation's practical feasibility in a series of empirical studies of its performance for a number of interesting challenges.

Chapter 6: Empirical Studies

In this section we present results from empirical studies aiming to assess the practical feasibility of the approach with respect to time. We believe that it is important to To the best of our knowledge, there is no established set of benchmarks for the type of reasoning presented in this dissertation, and so we have generated a set of novel problem instances that allow us to examine and make initial conclusions about the performance of the implementation on a number of interesting cases.

The first three experiments test the implementation on two types of automatically generated problem instances. In the first type of problem, N events occur simultaneously in the first step and cause N literals that do not hold in the first step to hold in the subsequent step. We refer to a set of causes occurring under these conditions as *fully concurrent*. In the second type of problem, N events occur over N steps (one per step), causing N literals that do not hold in the first step to hold by the $N + 1$'th step. We refer to these cases as *strict sequences*. We generate fully concurrent and strict sequence cases for direct and indirect causes, and for each case we want to explain how all N literals have been made to hold in the final step of the path. In these cases, all static chains of indirect causes are of length 1.

In the fourth and fifth experiments, we test the implementation on two additional types of automatically generated problem instances where static chains have length greater than or equal to 1. In the first type, one event occurs in the first step and indirectly causes a single literal to hold in in the next step via a static chain of length C. We refer to this case as a *single-source chain*. In the second type, N events occur simultaneously in the first state, each of which sets off a unique "chain reaction", the ramifications of the N events occurring together cause a single literal to hold in the subsequent state, and we refer to this case as a *multi-source* or *N-source chain*.

6.1 Full Concurrency and Strict Sequences

In the first group of experiments, we compare the time required to compute direct and indirect causes for increasingly large instances of fully concurrent and strict sequence cases.

6.1.1 Setup

Given a value N, $4N$ problem instances for this experiment are generated as follows:

1. *Fully Concurrent Direct Cause (FCDC)*: For every value in $\{1, \ldots, N\}$, a problem instance is generated with N events, N fluents, and one dynamic law for each of the N events, the heads of which are non-negated literals corresponding the the N fluents. In other words, each event directly causes a single literal to hold in the subsequent state. All fluents are negated in step 1 of the path. All N events occur at step 1 of the path, and all N fluents are non-negated in step 2. The outcome consists of the literals in step 2.

2. *Strict Sequence Direct Cause (SSDC)*: For every value in $\{1, \ldots, N\}$, a problem instance is generated in a similar way to FCDC instances, except that in this case the path consists of $N + 1$ steps, with the Nth event occurring at the Nth step. As before, all fluents are initially negated, and all have become non-negated by the $N + 1$th step, and the outcome again consists of the literals of the $N + 1$th step.

3. *Fully Concurrent Indirect Cause (FCIC)*: For every value in $\{1, \ldots, N\}$, a problem instance is generated with a similar structure to that of the *FCDC* problems. In this case, however, $2N$ fluents are created, half of which correspond to the direct effects of events the N events, and the other half of which correspond to indirect effects of those events. For every event N, there exists a dynamic law whose head is the precondition of a state constraint. The outcome contains the heads of all state constraints, and all $2N$ fluents are initially negated.

4. *Strict Sequence Indirect Cause (SSIC)*: For every value in $\{1, \ldots, N\}$, a problem instance is created with a similar structure to SSDC, with the addition of the characterization of indirect effects via dynamic laws and state constraints from FCIC cases.

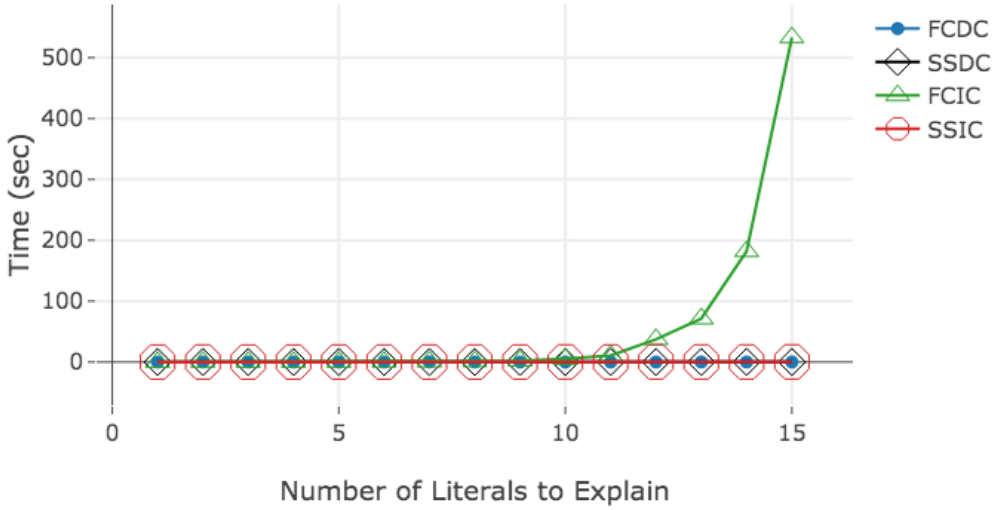

Figure 6.1: Comparing the time to explain up to and including 15 literals for all fully concurrent and strict sequence cases (direct and indirect)

6.1.2 Experiment 1

As a result of testing several values of N for Experiment 1, we find that the most insight is gained when examining the results when $N = 15$.

Results. The approach takes approximately 700 seconds to explain 15 literals caused by 15 concurrent indirect causes (FCIC), and less than 0.1 seconds to explain 15 literals for the remaining three cases (FCDC, SSDC, FCIC). This can be explained by the fact that in the FCIC case, $2^N = 2^{15} = 32,768$ sets need to be considered as potential indirect causes for each of the 15 literals, whereas in the SSIC case only one set needs to be considered as an indirect cause at each of the 15 steps. It also makes sense that the FCIC case overtakes both direct cause cases because testing for direct effects of events via dynamic laws that apply in a particular transition ostensibly requires less time than is needed generating and testing potential static chains that would explain indirect causation. This is likely because when looking for static chains, both dynamic laws *and* state constraints need to be considered to find the first link of a chain and then the heads of state constraints

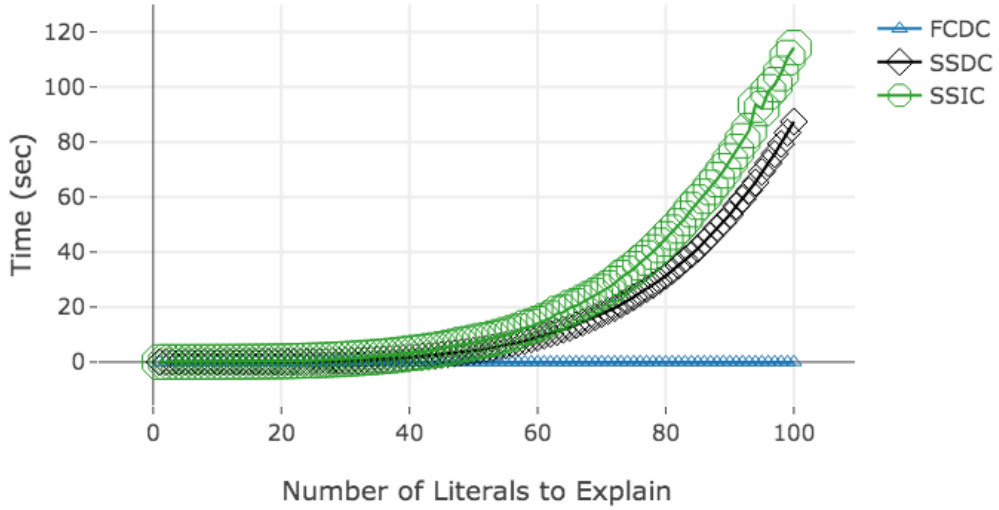

Figure 6.2: Comparing the time to explain up to and including 100 literals for fully concurrent direct cause (FCDC), single sequence direct cause (SSDC), and single sequence indirect cause (SSIC).

in the first link must tested for connections to the preconditions of additional state constraints until either an outcome literal is found (indicating that it has been indirectly caused) or until the space of state constraints has been exhausted.

6.1.3 Experiment 2

In the second experiment, we compare the time required to computed SSIC, FCDC, and SSDC cases with the goal of gaining insight into the relative performance of the approach on these cases for larger values of N. The setup for Experiment 2 is nearly identical to experiment 1, with the exclusion of problem instances for the FCIC case. Therefore, given a value N, $3N$ problem instances are created corresponding to the remaining three cases. As a result of testing several values of N for Experiment 2, we find that significant insight is gained when examining the results when $N = 100$.

Results. The approach takes approximately 114 seconds to explain 100 literals caused by 100 indirect causes occurring in a strict sequence (SSIC), approximately 87 seconds to explain 100 literals caused by 100 concurrent direct causes, and approximately 0.03 seconds to explain 100 literals

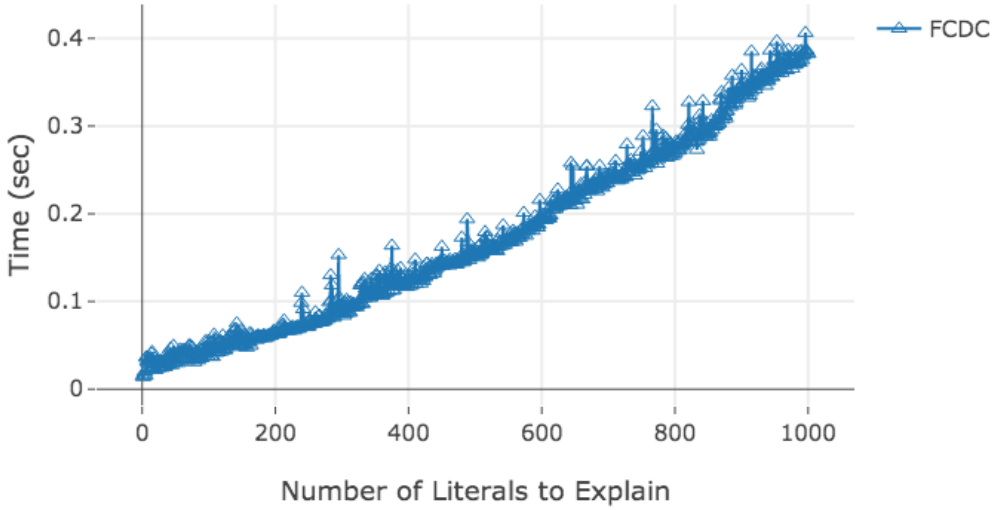

Figure 6.3: Observing the time required to explain up to and including 1000 literals for fully concurrent direct causes (FCDC).

caused by 100 direct causes occurring in a strict sequence. It makes sense that the SSIC case requires the most time of the three cases tested in this experiment because, as mentioned in the discussion of the previous experiment, it seems that generating and testing for static chains at each step of the path is a more computationally intensive task than testing just for direct effects via dynamic laws. We also see in Figure 6.2 that the time required to compute the SSDC case quickly overtakes the FCDC case in this experiment once the number of literals to explain exceeds 60. This can explained by the fact that in the SSDC case, each of the 100 dynamic laws needs to be tested 100 times for direct effects, one for each literal in the outcome, in each of the 100 steps of the generated path, even though obviously only one dynamic law will apply at each step. In the FCIC case, however, 100 dynamic laws are tested only *once* for each of the 100 literals in the outcome because there is only one step in the path. Considering these details, it is not surprising that the time required to compute the SSDC case is greater than the time needed to compute the FCDC case for Experiment 2. We also see little to no change in time to compute the fully concurrent direct cause case in the 1 to 100 range, a point which is addressed in Experiment 3.

6.1.4 Experiment 3

The third experiment is motivated by the insight from the previous experiment that the approach takes less than 1 second to compute the fully concurrent direct cause case for up to 100 literals. In this experiment, we vary N from 1 to 1000 to get a better idea of how the performance changes for problem instances with even more causes and literals to consider. For Experiment, given a value N, N problem instances are created corresponding to the FCDC case. As a result of testing several values of N for Experiment 3, we find that insight is gained when examining the results when $N = 1000$.

Results. The approach requires approximately 0.4 seconds to explain 1000 literals caused by 1000 concurrent direct causes. We can see that this is the least challenging case to compute in this series of experiments, seemingly due to the fact that each of the 1000 dynamic laws only needs to be tested once for direct effects for each of the 1000 literals that need to be explained.

6.2 Single-Source Chains

The fourth experiment departs from analysis of fully concurrent and strict sequence cases and directs focus to the static chain cases presented in the introduction of this section. Here, we look at the time taken to explain indirect cause for single-source chains of length 1 through 1000.

6.2.1 Experiment 4

Given a value C, C problem instances for this experiment are generated as follows. For every value in $\{1, \ldots, C\}$, a problem instance is generated with 1 event and C initially false fluents. 1 dynamic law is created for the single event, the head of which is the non-negated literal corresponding the first of the C fluents. For every $i \in \{1, \ldots, C\}$, there exists a state constraint linking the ith fluent in C to the $i+1$th fluent (i.e., $i+1$ **if** i), and the outcome contains only the Cth fluent.

Results. The approach requires approximately 50 seconds required to explain a single literal that was indirectly caused via a chain of events with 1000 links. Conclusions about the static chain, and therefore indirect causation, are reached by first considering the direct effect of the single event e and then testing each of the 1000 state constraints to see if any have become satisfied as a "side-

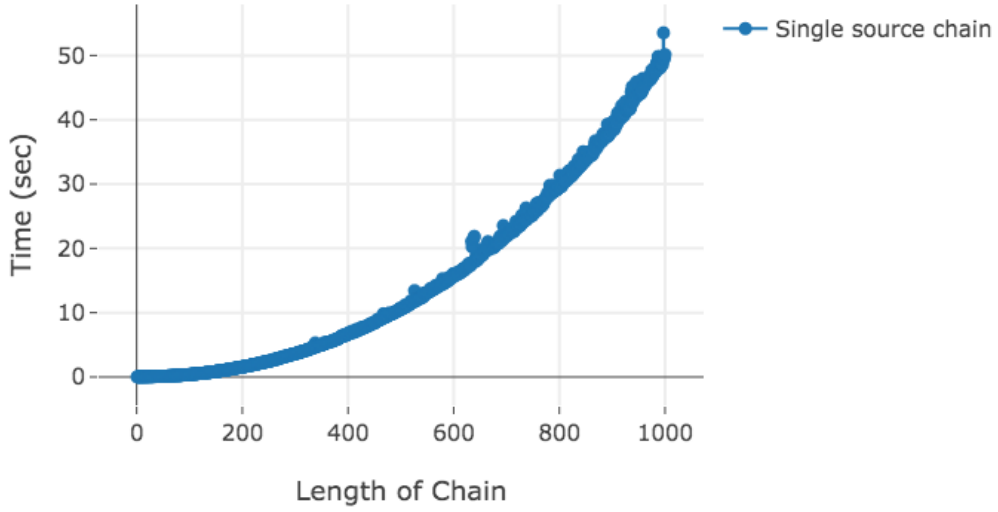

Figure 6.4: Observing the time required to explain up to and including 1000 literals for single-source chains ranging in length from 1 to 1000 links.

effect" of e. From here, it needs to be determined whether or not the preconditions of any of the 1000 state constraints have become satisfied due to the consequence of a state constraint that has become satisfied as an indirect effect of e, and in our case this process will occur 1000 times before the outcome literal is finally linked to e's direct effect. It is easy to imagine that the time needed to compute causes for long static chains in more complex scenarios would increase dramatically, but we believe that this experiment provides reasonable insight into the baseline performance of the approach for long chains of indirect causation in the context of our framework. Next, we will present an example of a more complicated scenario involving static chains.

6.3 Multi-Source Chains

In the final experiment, we examine the time required to compute multi-source chains with an increasingly large set of *source events* and increasingly long static chains.

6.3.1 Experiment 5

Given values C and N, $C * N$ problem instances for this experiment are generated as follows. For every value i in $\{1, \ldots, C\}$ and value j in $\{1, \ldots, N\}$, an instance contains $i * j + 1$ fluents, all of which are negated in the initial state. The instance also includes j events, and one dynamic law for each event, the head of which is a non-negated literal. For each event e, there also exist i state constraints that create a static chain from e to a unique literal l. Finally, each unique l is a member of the set of preconditions for a final state constraint, the head of which is a literal l' which is not in a precondition for any law and is the only member of the outcome. It may be convenient to think of this problem instance as a set of distinct chain reactions (paths) through a state that only together indirectly cause a literal to hold in that state.

Results. The approach requires approximately 1.2 seconds to compute a 10-source chain of length 1 and approximately 50 seconds to compute a 10-source chain of length 10. Something interesting to note here is that even though $2^{10} = 1,024$ sets need to be considered for the 10-source chains, the chain of length 1 takes significantly less time to explain than the FCIC case that considers the same number of sets (approximately 7 seconds to consider 10 simultaneous events). This is likely due at least in part to the fact that in this case, we are explaining a single literal rather than 10, which drastically reduces the number of tests that need to be made to explain the entire outcome. On a related note, we see that the time to compute a 10-source chain of length 10 is significantly greater than the time needed to compute the FCIC case at 10, by a factor of approximately 7. This is likely due to the fact that the reasoner has to consider a larger number of state constraints in this case, $N * C = 10 * 10 = 100$ rather than $N = 10$ for the fully concurrent indirect cause case.

6.4 Conclusions

In these studies, we have examined the performance of the approach under a variety of interesting conditions and have reached some initial conclusions about the feasibility of the approach. Experiments 1, 2, and 3 allow us to conclude that explaining multiple indirectly caused literals that have become true as the result of a single transition is significantly more challenging than reason-

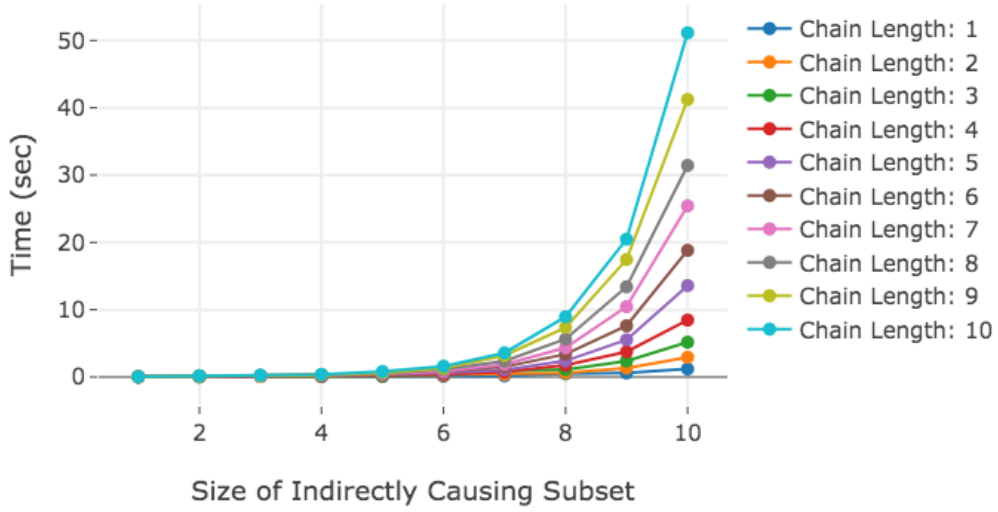

Figure 6.5: Observing the time required to explain one literal for N-source chains range in length from 1 to 10 links, where N ranges from 1 to 10.

ing about a similar scenario with direct causation. The insights gained from Experiments 4 and 5 indicate that there is a vast space of cases of indirect cause that can be represented and reasoned about.

Although an exhaustive study of the entire space of possibilities is needed before general claims have been made, we believe that these experiments show that the approach is promising. We have demonstrated that the approach is able to explain causation in scenarios that clearly far exceed the complexity of the toy examples from the literature, and although in certain challenging cases the time needed to compute the causes is not nominal, the approach is able to reason arguably faster that a human given the same problems and reasoning framework. We posit that an interesting open problem is to investigate the enumeration of the space of scenario types, including but not limited to cases of varying N values for multiple direct and/or indirect causes in a given scenario, multi-source chains with direct and indirect causes, and scenarios that combine all of the types of cases that we have examined here towards the creation of novel and challenging benchmark datasets and compelling use cases inspired by complex real-world scenarios.

Chapter 7: Related Work

In this Chapter, we will cover a selection of related work in the area of causality. First, we will address the most well-known and widely studied approach in the field, the original work on the counterfactual evaluation of structural causal models that sparked the initial interest in actual cause within AI. We will discuss the limitations of the original approach, as well as attempts to amend and improve the work. Next, we will present and discuss the approach that was the most influential to the dissertation work which took an important step towards modeling a scenario as a sequence of events that forces state to evolve. Following that, we will present and discuss the work that is most closely related to ours, which leverages Situation Calculus and its solution to the frame problem to identify the "achievement" of conditions expressed in first-order logic for a given situation of interest.

7.1 HP Account of Actual Causation

The structural causal model of [54] is one of the most well-known and influential approaches to representing and reasoning about causal relationships among variables. Here we present a definition of structural causal models (SCM) and the HP approach to defining actual cause for SCMs.

Structural Causal Models.

An SCM is a triple $M = \langle U, V, F \rangle$ where:

- U is a set of background or exogenous random variables (RVs) that are determined by factors outside of the model. No explanatory mechanisms exist in M for U.
- V is a set $\{V_1, V_2, ..., V_n\}$ of endogenous RVs that are determined by variables inside the model.
- F is a set of functions that maps $U \cup (V \backslash V_i)$ to V_i. In words, F determines how the exogenous and endogenous RVs affect the values of a specific endogenous RV in V.

A function in F determining the value of an RV X takes the form $X := f(\vec{Y})$ which states that a cause of X taking on value x is an assignment$\vec{Y} = \vec{y}$ of a vector of non-X RVs in M such that $f(\vec{y}) = x$. In this account, the notation $\vec{X} = \vec{x}$ is shorthand for the conjunction of RV settings $X_1 = x_1 \wedge \ldots X_n = x_n$. Note that the definition of functions in F requires that the equations are acyclic by excluding X from the set of variables that can influence the value taken on by X. Therefore, an assignment of values to exogenous RVs U determines a single possible setting of values for RVs in V. For cases of actual causation, we can restrict the domain of RVs to $\{t, f\}$ so that they can be considered as propositional symbols [21]. This provides a convenient way to represent RVs as ground atoms for evaluation against logic programming approaches to representing and reasoning about actual cause.

Actual Cause.

In this section we reproduce the HP account of actual cause given in [3]. Here, M is a structural causal model with endogenous RVs $\mathcal{V}$, $\vec{U} = \vec{u}$ is a setting of exogenous RVs, $\vec{X} = \vec{x}$ is a setting of RVs in M, and ϕ is a boolean formula of RVs. The notation $(M, \vec{u}) \models \psi$ means that ψ is true in $(M, \vec{u})$. The notation $(M, \vec{u}) \models [\vec{X} \leftarrow \vec{x}]\phi$ means that ϕ holds in $(M, \vec{u})$ after setting RV vector $\vec{X}$ to $\vec{x}$.

Definition 12. *(HP account of actual causation)*

$\vec{X} = \vec{x}$ is an actual cause of ϕ if the following three conditions hold:

AC1. $(M, \vec{u}) \models (\vec{X} = \vec{x}) \wedge \phi$.

AC2. *There exists a partition $(\vec{Z}, \vec{W})$ of $\mathcal{V}$ with $\vec{X} \subseteq \vec{Z}$ and some setting $(\vec{x'}, \vec{w'})$ of the variables in $(\vec{X}, \vec{W})$ such that if $(M, \vec{u}) \models Z = z*$ for $Z \in \vec{Z}$, then:*

a. $(M, \vec{u}) \models [\vec{X'} \leftarrow \vec{x}, \vec{W'} \leftarrow \vec{w'}]\neg\phi$.

b. $(M, \vec{u}) \models [\vec{X'} \leftarrow \vec{x}, \vec{W'} \leftarrow \vec{w'}, \vec{Z'} \leftarrow \vec{z'}]\phi$ *for all subsets $\vec{W'}$ of $\vec{W}$ and all subsets $\vec{Z'}$ of $\vec{Z}$.*

AC3. *$\vec{X}$ is minimal; no subset of $\vec{X}$ satisfies satisfies conditions* AC1 *and* AC2.

Condition AC1 states that both $\vec{X} = \vec{x}$ and ϕ are true in the actual world. Condition AC2.a requires that changing $(\vec{X}, \vec{W})$ from $(\vec{x}, \vec{w})$ to $(\vec{x'}, \vec{w'})$ changes ϕ from true to false. Condition AC2.b

states that setting any subset of variables in $\vec{W}$ to their values in $\vec{w}$ should have no effect on ϕ, as long as $\vec{X}$ is kept at its current value $\vec{x}$, even if all the variables in an arbitrary subset of $\vec{Z}$ are set to their original values in $\vec{u}$. The minimality condition AC3 ensures that only these elements of the conjunction $\vec{X} = \vec{x}$ are essential for changing ϕ in AC2(a) are considered part of the cause.

7.1.1 Discussion

The HP approach was a milestone for reasoning about actual causation in the context of Artificial Intelligence, using structural equations and SCMs to model and reason about causal relationships among variables. In the context of actual causation, structural equations work well for the kind of "toy example" typically considered in the literature, and the approach indeed allows us to unearth numerous sophisticated causation phenomena. However, we do not believe that it provides a suitable framework in which to reason certain advanced scenarios, including the behavior of complex cyber-physical systems. This may be due, at least in part, to a lack of distinction between states and events (or actions) that force state to evolve over time. In [13], the authors attempt to address these limitations by modeling HP causality and counterfactuals in the situation calculus; however, the approach requires the modeler to arbitrarily declare some variables as *transitional* and others as *enduring*, where the former are meant to represent events and the latter represent fluents. Moreover, the approach does not require that situations are executable, which can result in a paradox for cases of dismissed preconditions, as discussed in [20].

Another limitation of HP, and related approaches, is in the use of the counterfactual definition of actual cause. As we stated in Chapter 1, this definition is inspired by the intuition that if X caused Y, then not Y if not X [10, 58], and it has been pursued as a condition of actual causation in numerous works [8, 9, 13, 59–61]. We have already discussed how the counterfactual criteria can fail to recognize causation in a number of practical cases such as overdetermination, preemption, and contributory cause [11, 12, 62]. Here we discuss these challenges to counterfactual definitions of cause in greater detail, and point out how our approach handles these cases by referring to the running example from Chapter 3.

In cases of overdetermination, removing one of the multiple sufficient causes from the scenario

will not prevent the outcome from occurring. Therefore, if X and Y are both sufficient to cause Z, the counterfactual definition of cause may not identify X or Y as an actual cause because removing one or the other will not prevent Z. In our running example, e_1 and e_2 occurred simultaneously in the first state and each was sufficient to cause A regardless of the other event. Both were identified as direct causes because both satisfied the definition of direct cause.

In cases of preemption, multiple sufficient causes have occurred, but the first cause to bring about the effect preempts the potential effects of any other cause. The bottle breaking problem presented in Chapter 4 is a well-known example of preemption from the literature. Counterfactual definitions of actual cause will have trouble with this example because the bottle shattering is not counterfactually dependent on either throw. In our extended example in Chapter 4, we correctly identified that Suzy's throw was a direct cause of the bottle breaking. In the running example, e_5 occurred but was unable to cause C because e_3 had already done so. Only e_3 satisfied the definition of direct cause. Note that sometimes a distinction is made between late and early preemption. An example of the former is that Suzy's rock reaches the bottle before Billy's. An example of the latter is Billy not throwing because Suzy has already thrown. We also introduced the notion of single-transition preemption in Chapter 3, which posed a problem for the simple notion of indirect cause, but is handled intuitively by the improved definition.

Finally, recall that contributory causes are causes which only together are sufficient to cause an outcome. If X and Y are contributory causes for Z, the counterfactual definition of cause could identify X and Y as actual causes because removing either of them will prevent the outcome, but it can not tell us that X and Y must occur *together* in order to bring about Z. Recall that in our example, e_4 and e_5 had to occur together to indirectly cause D to hold. We also saw an interesting example of contributory cause in the Firing Squad example in Chapter (Examples).

More recent approaches such as [13, 14, 21] have addressed some of the shortcomings associated with the counterfactual criterion by modifying the existing definitions of actual cause or by modeling change over time with some improved results. However, there is still no widely agreed upon counterfactual definition of actual cause in spite of a considerably large body of work aiming

to find one.

We have attempted to map the running example from Chapter 3 to a structural causal model and found that the translation of the action description is not straightforward. Given the problem $\psi_E = \langle \theta_E, \rho_E, AD_E \rangle$, it is conceivable that it may be possible to emulate this type of approach by representing AD_E as a *Causal Bayes network* (CBN) AD_{CBN}, and iterating through the events $v_i \in v$ to evaluate each statement *"If v_i had not been true, then θ would not be true"*. A CBN is a probabilistic directed acyclic graph modeling causal relationships among random variables (nodes). A possible approach to translating causal laws of AD to AD_{CBN}, is to add a node for every event and fluent in AD and then to add a directed edge from node x to y when there exists a causal law such that x causes y. However, several challenges arise in such a mapping, making the translation itself non-trivial. The most notable involves translating laws in which a fluent f occurs, possibly negated, both in the precondition and in its consequence. Consider this relatively simple law from AD_E:

$$e_1 \text{ causes } A \text{ if } \neg A$$

This law states that e_1 can cause A to hold if it does not already hold. The translation approach we outlined would require introducing two random variables, $X = e_1$ and $Y = A$, where the values of X and Y are 1 if they are true and 0 if they are false. Modeling the fact that the value of X affects the value of Y is straightforward by adding a directed edge from X to Y, but it is not immediately obvious how to model the fact that Y's value also affects X's ability to take on the value 1. Adding a directed edge from Y back to X violates the structure of a CBN by adding a cycle to the graph, and omitting the link results in loss of commonsense knowledge and extending the definition of CBNs to account for such a relationship, e.g., by parameterizing fluents with their time step, seems non-trivial.

7.2 CP-logic Account of Actual Causation

In [21], the author identifies that a shortcoming of the definition of actual cause for SCMs is the lack of accounting for the dynamic nature of stories – the HP definition considers only the final state of

the world after all of the events have occurred and does not model how the state of the world changes over time in response to events of a story. Vennekens claims that Shafer's probability trees [63] are more suitable for representing the dynamic nature of causal relations. A Shaferian probability tree is a structure whose nodes represent states of the world and whose edges represent transitions between them. [21] uses CP-logic to provide a syntactic representation of Shaferian trees and defines actual cause for this structure.

The general form of a CP-law is as follows:

$$\forall \overrightarrow{x} \, (A_q \, : \, \alpha_1) \vee \ldots \wedge (A_n \, : \, \alpha_n) \leftarrow \phi$$

where ϕ is a first-order formula, A_i are atoms, $\overrightarrow{x}$ contains all free variables of ϕ and A_i, and α_i's are non-zero probabilities such that $\sum \alpha_i \leq 1$. A CP-law states that logical formula ϕ causes a non-deterministic event for which each A_i is a possible outcome with probability α_i. A CP-law with a deterministic effect is denoted by $A \leftarrow \phi$.

A collection of CP-laws is called a CP-theory $\mathcal{T}$ and describes the non-deterministic evolution of a domain and is formally represented by a Shaferian probability tree called an *execution model*. A branch $b = (s_0, \ldots, s_n)$ of an execution model corresponds to a specific sequence of events that occur in a story, where $s_0, \ldots, s_n$ are states of the world each containing all atoms in T and their values in that state. The values of all atoms in the initial state s_0 are set to false and the occurrence of events causes RVs to switch from false to true in subsequent states. The event that causes a transition between states s_1 and s_2 is given by $\mathcal{E}(s_1)$ which is the CP-law that caused a transition between s_1 and s_2.

Under the CP-logic account, if one wants to determine if atom C is an actual cause of atom E using a counterfactual approach, they begin by modifying the CP-Theory $\mathcal{T}$ so that there is a zero probability that C will occur in the execution model of $\mathcal{T}$. This is done by performing a transformation on each CP-law r of $\mathcal{T}$ to ensure that C does not belong to the $head(r)$. The CP-theory resulting from this transformation is $T^{\neg C}$.

In the interest of clarity of the presentation, we give a simplified interpretation of Vennekens'

account of actual cause. The omitted details of the definition, as well as a definition of actual cause in an incomplete information setting, are described by the author in [21].

Definition 13. *(CP-logic account of actual causation)*
Given a CP-theory $\mathcal{T}$, let $b = (s_0, \ldots, s_n)$ be a branch of $\mathcal{T}$'s execution model. Let C and E be atoms that both hold in the final state s_n of b. C is an actual cause of E in branch b if C does not occur in the execution model of $\neg C$.

In other words, C is an actual cause of E if preventing C's occurrence in the story would cause E to no longer hold. The definition enables the authors to handle preemption in the "blindingly obvious" way, which is certainly quite intuitive: whichever event C was the first to produce the outcome C is the actual cause and anything that happens after the fact doesn't count. This approach captures the notion that it is not possible to cause something that has already occurred.

7.2.1 Discussion

As we have already stated, this approach is closely related to the dissertation work. The concise handling of preemption in the approach strongly influenced our investigation into examining intrinsic causal mechanisms contained in a path, and we handle preemption in a similar way. However, there are two significant differences between the frameworks. The first is the lack of distinction between events and the state of the world, which by now we believe we have demonstrated to be an important distinction to make when modeling scenarios and reasoning about actual cause. The language consists of a single type of law that is not sensitive to the types of objects it relates. The second difference is the CP-logic approach's dependence on the counterfactual definition of cause, the drawbacks of which we have already enumerated[1].

In a turn from relying on counterfactual dependence alone, a more recent approach along this line of investigation [9, 50] propose a principled approach to defining and analyzing actual causation starting from concepts of counterfactual dependence, contribution, and production, positing that definitions of actual cause could be built from a deeper understanding of the relationships among these concepts as first principles. An even more recent approach [16] introduces a new

[1] Preemption excluded, in this case.

language related to CP-Logic in which causal mechanisms and causal processes are represented in formal logic, allowing the authors to define concepts of causal production for building a framework of actual causation. While there is an obvious relationship between these more recent approaches and the approach of the dissertation, the nature of this relationship is not yet clear and this analysis is left for future work, likely in collaboration with the authors.

7.3 Situation Calculus Semantics for Actual Causality

In one of the most related known research to the dissertation work, the authors of [19] depart from a strictly counterfactual approach, using a similar insight to our own that actual causation can be determined by reasoning about what has actually happened in a given scenario rather than hypothesizing about counterfactual worlds, at least in finding what caused an outcome to "appear" in a scenario. Leveraging the Situation Calculus (SC) to formalize knowledge, the approach aims to identify a single event that has caused an SC formula φ to become true in a situation, and then uses a single-step regression approach to identify a list of events deemed relevant to the event's ability to realize φ.

In SC, a dynamic world is modeled as a progression through *situations* in response to actions being performed in the world. SC was originally proposed in [39] and extended by [64], an important contribution of which was a proposed solution to the frame problem discussed in Chapter 2. The latter of the two is the formalism of choice for [19] because, as they state, it allows them to take advantage of some of the findings of the earlier cited investigation by [13], in which the authors attempted to redefine counterfactuals in the context of SC. In the interest of clarity of the presentation, we will provide sufficient background of SC to give an overview of the causal reasoning approach, but detailed presentations of the approach and SC as it pertains to it are described in [19].

In the SC formalism, a constant S_0 denotes the *initial situation* and represents an empty list of actions. A *situation term* $do([\alpha_1, \ldots, \alpha_n], S_0)$ is the situation that results from executing actions $\alpha_1, \ldots, \alpha_n$ consecutively. When none of the action terms have variables, then the situation term is an (actual) narrative. A basic action theory (BAT) $\mathcal{D}$ includes axioms $\mathcal{D}_{S_0}$ describing the initial

situation, action precondition axioms $\mathcal{D}_{S_0}$ stating when and action a can be executed in a situation s, as well as a successor state axiom (SSA) for each fluent F in the model which capture the "non-effects" of actions. BAT $\mathcal{D}$ additionally contains auxiliary axioms including abbreviations $executable(s)$ indicating that every action in s was executable in s, as well as state constraints that represent conditions that are true in every state. The basic computational challenge of SC, called the *projection problem*, is the task of determining if a BAT entails $\varphi(\sigma)$ for a ground situation term σ, and $\varphi(\sigma)$ is a logical formula that is true, or *uniform*, in σ.

[19] builds upon this foundation to define and reason about actual cause. A scenario is represented by a *causal setting*, a triple $\langle \mathcal{D}, \sigma, \varphi(\sigma) \rangle$ where $\mathcal{D}$ is a BAT, σ is a ground situation term such that the every action in σ executable in the situation, and $\varphi(\sigma)$ is a formula uniform in σ. The authors claim that if an action α of the action sequence triggers the formula $\varphi(\sigma)$ to change its truth value from *false* to *true* and if there is no action in σ after α that changes the formula's value back to *false*, then α is an actual cause of achieving $\varphi(\sigma)$ in σ). The authors formally define α as a *primary achievement cause* of $\varphi(\sigma)$ with respect to the causal setting.

The authors claim that this event is not sufficient to explain how $\varphi(\sigma)$ became true, as other actions in σ occurring before α may have played a role in the incremental "build up" to the eventual achievement of the $\varphi(\sigma)$. The authors then use a single-step regression operator to reason backward over the scenario, yielding a new formula $\varphi'(\sigma)$ (and corresponding causal setting) that enabled α's causing of $\varphi(\sigma)$. An achievement cause α' in σ may be found for $\varphi'(\sigma)$, and is deemed part of the actual cause of $\varphi(\sigma)$. The approach is repeated until the beginning of the scenario has been reached at which point no further causal settings are yielded and the resulting set of actions is referred to as the *achievement causal chain*.

7.3.1 Discussion

It is easy to see that there is a clear conceptual relationship between the notion of achievement and the transition states of the dissertation framework. However, the underlying notions of actual cause and the choice of SC as a formalism lead to several notable differences with the dissertation work.

There are two primary differences in the definitions of actual cause. First, the authors claim that a formula has been caused only if it has not later been "un-caused" by a subsequent event. This implies that they are asking *"Why did X hold at the end of the scenario?"* rather than *"What caused X to hold in the scenario"*? This is an important distinction, and we believe that they impose an unnecessary condition on the achievement causal chain that might result in non-intuitive results. For example, consider the following action description AD:

$$e_1 \textbf{ causes } A \tag{7.1}$$

$$e_2 \textbf{ causes } \neg A \tag{7.2}$$

Consider now a path $\rho = \langle \sigma_1, \epsilon_1, \sigma_2, \epsilon_2, \sigma_3 \rangle$ in which $\neg A \in \sigma_1$, $\epsilon_1 = \{e_1\}$, and $\epsilon_2 = \{e_2\}$. Clearly, $A \in \sigma_2$ and $\neg A \in \sigma_3$. If the outcome of interest is $\theta = A$. We claim that we should identify e_1 as a direct cause of A for transition state σ_2 of θ in ρ, even though the outcome was no longer satisfied in the subsequent state. Mapping this example to the approach of [19], it seems that A would not be considered to be caused at all because it did not hold at the end of the scenario.

The second important difference is the notion of the achievement causal chain. In [19], they define actual cause in terms of every event that helped to "set the stage" for an action to cause a formula to change its truth value from false to true. This means that every single event that had any influence on causation is part of the cause. Although this can be a nice feature for simple examples, we find that in practical settings this definition casts too wide a net and surely would return events that are of no interest to the user asking for a causal explanation. In fact, the subject of *transitivity* as a general condition for cause is still being examined [50, 65], and several counterexamples have been examined that demonstrate transitivity should not always be considered when reasoning about actual causation (see, e.g., [50, 54]). We have demonstrated over multiple examples in Chapter 4 that our definitions can be used to uncover deeper aspects of a causal mechanism *at will*, rather than identifying every event that contributed in any way to causing an outcome of interest to hold. However, since there are certainly cases in which automating the deeper reasoning approach

would be desirable, it may be possible to identify notions of *supporting causes* defined in terms of the preconditions of the dynamic laws, state constraints, and executability conditions of an action description, as we have shown in [53].

A related work by the same authors [20] also introduces the notion of a *maintenance cause*, which "protects" a previously achieved effect from potential causes that could have "destroyed" the effect. This consideration differs from our approach, but is an important consideration and an interesting direction for future work. We propose that it could be possible to extend our approach to perform a type of counterfactual reasoning over the paths to achieve a similar definition, perhaps leveraging insights from our initial investigation into defining notions of cause in the context of $\mathcal{AL}$[66].

The choice of SC as a formalism also results in some notable differences in the technical approach and reasoning capabilities. In Reiter's SC, a *situation* is a finite sequence of actions, which is notably contrary to the original definition of [39] which considers states. Reiter states in [67]:

> *"A situation is a finite sequence of actions. Period. It's not a state, it's not a snapshot, it's a history."*

Examining and analyzing Reiter's choice of representation is beyond the scope of the dissertation work and will not be discussed in any great detail here, however it is important to note that we consider this point of view and the resulting formalism to be a significant underlying drawback to the causal reasoning approach from the outset.

Consider the position that philosophical and mathematical accounts of actual causation aim to take steps towards emulating the level of intuition used by humans when reasoning about the actual cause of an outcome in a given scenario. In addition to retaining the sequence of events that have occurred, we often employ an understanding of the context in which the events have occurred to form a picture of how the state of the world has changed as a result. In this way, we gain a deeper understanding of the meaning of the events and their causal influence. We posit that paths of an action description, which include complete descriptions of events in each state as well as the events that have occurred, map more naturally to human intuition as we have proposed it here than simply a sequence of the events. Under this consideration, Reiter's position that a history consists only of

actions seems incomplete[2]. This point is arguable, however, it is our position that an account of actual causation that is intended for the benefit of humans should reflect our reasoning approaches as clearly as possible so that its results can be consumed in a straightforward and intuitive way.

Another important distinction between the choices of formalism is that SC allows only for the concurrent execution of singular actions, whereas the semantics of $\mathcal{AL}$ permits the co-occurrence of events. In reality, events occur simultaneously and this definition limits Batusov's ability to model such a situation. Although concurrency of actions has been explored for SC [23, 68], the authors state that it is not a feature of their approach and it is not immediately clear how the definitions should be extended to enable such support.

Compared to $\mathcal{AL}$ formalizations, SC formalizations also incur limitations when it comes to the representations of indirect effects of actions, which play an important role in our work, and the elaboration tolerance of the formalization. Additionally, SC relies on First-Order Logic, while $\mathcal{AL}$ features an independent and arguably simpler semantics.

7.4 Additional Approaches

Finally, it should be noted that a number of other interesting approaches exist linking causality and non-monotonic reasoning. We discuss one in some detail here and provide some closing discussion on research directions relating these topics.

7.4.1 Causal Logic Programming

Another interesting, although less related, approach to reasoning about cause in logic programming is given by [69]. This work presents a causal extension of logic programming under stable model semantics which enables the representation of alternate causes of each true atom in the solution of a logic program. These representations are called *causal proofs* which are trees encoding the ordered application of logic rules that resulted in the truth (or falsehood) of the atom in question. The trees are obtained by adding labels to rules. Consider the Yale Shooting Scenario of [52]:

There is a turkey called Fred and shooting a loaded gun will kill it. Suzy loads the gun

[2]Imagine if history books were simply timelines.

and then shoots.

This scenario can be modeled as follows using causal logic programming:

$$r1 : dead \leftarrow shoot,\ loaded \qquad suzy_1 : load$$

$$r2 : loaded \leftarrow load \qquad suzy_2 : shoot$$

where the symbols on the left of the colons are the rule labels. While we will not go into the details of this approach here, the Yale Shooting program will result in the atom $dead$ being true, and the following is the causal proof showing how this atom was derived.

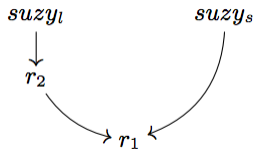

Figure 7.1: Causal proof of atom $dead$ for Yale Shooting program

This work is extended by [70] in which the author introduces *causal literals* which "inspects" the causal proofs in order to enable further reasoning. The author adds the following rule to the Yale Shooting program that corresponds to the final sentence of the elaborated scenario:

$$r_3 \ : \ prison \leftarrow (suzy :: dead)$$

Here, $(suzy :: dead)$ is a causal literal which holds when the $suzy$ actions $suzy_1$ and $suzy_2$ have been determined to be sufficient causes of $dead$ by inspecting the causal proof. As we can see in Figure 7.1, the atoms $suzy_1$ and $suzy_2$ together have been sufficient to cause r_1 to fire, making $dead$ true.

7.4.2 Discussion

While this work is somewhat less related to the dissertation work than the first two works covered in this section, there is a clear relationship in that this approach creates partial paths as we have defined them here. Consider again Figure 7.1. We consider that it may be possible to link causal proofs to the paths of our framework which can lead to interesting synergies yet to be explored.

There are numerous additional research efforts in this area, with varying goals (e.g. encoding the HP account via Logic Programming approaches [61, 71], explaining answer sets of ASP programs [72, 73], and reasoning about causal information [70, 74–77]) in the context of non-monotonic reasoning. Research relating these topics is steadily advancing, prompting interdisciplinary discussion and exploration of the role and placement of causal reasoning and LP in the landscape of modern computer theory and the software industry.

Chapter 8: Conclusions and Future Work

The result this dissertation work is a theoretical framework for reasoning about actual cause that leverages techniques from reasoning about actions and change, and an implementation of the framework via ASP that is sound and complete. We have demonstrated that framework leverages intuitive and iconic representations of scenarios as the evolution of state over time in response to events. The framework can be used to reason about direct and indirect causal influence of events over the state of the world, and can overcome traditionally challenging cases of actual cause that involve overdetermination, preemption, and contributory causation. We have shown that the framework identifies intuitive answers to a number of classic examples from the literature, as well as for a novel scenario in which reports from independently acting modules of a self-driving car are reasoned over to assign blame for unexpected behavior in a crash scenario. We have also demonstrated that the implementation of the framework is able to explain causation in scenarios that far exceed the complexity of the toy examples from the literature. Although in certain challenging cases the time needed to compute the causes is not nominal, the implementation reasons about the scenarios arguably faster that a human could given the same problems and reasoning framework.

8.1 Open Problems

Recall from our discussion in Chapter 7 that, in our examples, we have relied on human reasoning to identify the preconditions of events to create new problems that can be used to identify events that "set the stage" for a causing event of an outcome. As we discussed in Chapter 7, it can be desirable to have control over which preconditions of which events we would like to explain. It is easy to imagine, however, that in a complicated scenario that spans a great deal of time, a human would not want a causal explanation of every single event that contributed to a given outcome as likely not every event is directly relevant to the outcome. As we stated in that discussion, we believe that it is possible to identify notions of *supporting causes* defined in terms of the preconditions of

the dynamic laws, state constraints, and executability conditions of an action description, as we explored in [53] for a simpler version of the framework which did not support concurrent events.

For example, given a dynamic causal law λ in AD of form e **causes** l_0 **if** $\{l_1, \ldots, l_n\}$, let $e(\lambda) = e$, $c(\lambda) = l_0$, and $p(\lambda) = \{l_1, \ldots, l_n\}$. We can use a similar representation for executability conditions and state constraints, and can then introduce a set of definitions using which preconditions can be "extracted" from these laws. For instance in our running example, the literals $\neg A$ would be in a set $prec(e_1, \rho_E)$ of preconditions of law (3.1). In such a case new outcomes of interest can be constructed from the preconditions of laws in order to uncover additional information about the actual causal dynamics in a path.

We have also identified a case in which our framework identifies partially counter-intuitive subsets of elementary events as indirect causes. Consider the following action description:

$$e_1 \textbf{ causes } d_1 \tag{8.1}$$

$$e_2 \textbf{ causes } d_2 \tag{8.2}$$

$$i_1 \textbf{ if } d_1 \tag{8.3}$$

$$i_2 \textbf{ if } d_2 \tag{8.4}$$

Consider now a transition $\langle \sigma, \epsilon, \sigma' \rangle$ for which $\sigma = \{\neg d_1, \neg d_2, \neg i_1, \neg i_2\}$, $\epsilon = \{e_1, e_2\}$, and $\sigma' = \{d_1, d_2, i_1, i_2\}$. Let the outcome of interest be $\theta = \{i_1\}$. Due to the definitions of static chains and the improved definition of indirect cause, both $\{e_1\}$ and $\{e_1, e_2\}$ will be identified as indirect causes of d_1, even though intuitively one might expect only e_1 to be an indirect cause.

To understand how the two indirect causes are derived, recall that by Definition 9 of static chains, a state constraint s is a member of a link in a static chain $\chi(\epsilon, l, \sigma')$ if its preconditions are caused to be satisfied in σ' in part by the direct effects of one or more elementary events of ϵ. Therefore, the static chain $\chi(\{e_1, e_2\}, l, \sigma')$ will consist of a single link γ_1 containing laws (8.3) and (8.4), and $i_1 \in C(\gamma_1)$. Considering now Definition 10, condition 2 is not sufficient to filter out the extraneous event e_2 in this case because e_2's direct effect is a member of $P(\gamma_1)$. However, we

speculate that the definition of static chain can be expanded by reasoning *backward* from every state constraint s for which the literal of interest is the consequence, and whose preconditions hold in the state of interest. We propose that we then build the static chain by linking the consequences and preconditions of state constraints in an action description until one or more dynamic laws are found whose consequence initiated the chain of indirect causing. Considering again our new counterexample, we speculate that we would only identify $\{e_1\}$ as an indirect cause of d_1 because the state constraint (8.3) is the only law in the action description for which i_1 is its consequence, and (8.3)'s preconditions can be linked to the dynamic law (8.1) via its consequence d_1. Since the consequence of (8.1) is e_1, it would be an indirect cause of d_1. We also believe that it may be possible to leverage the causal process representation language of [16] to enable reasoning about the causal process in a state of interest.

8.2 Future Work

In addition to addressing the open problems discussed in the previous section, we believe that several compelling research directions stem from this dissertation.

There is work to be done in extending the representation capabilities of the framework. While the language $\mathcal{AL}$ allows us to elegantly describe the direct and indirect effects of events in a dynamic domain, it does not naturally support representations of certain realistic cases in causation such as direct contributory causes, time delayed effects, non-deterministic actions and cases in which *events* are triggered by other events and/or circumstances. An important next step along these lines will involve exploring generalization of the framework to any representation language that describes a transition diagram, and in which one can identify direct causes, state constraints, preconditions, and consequences.

Along the lines of exploring different types of causes, one type of causation that the framework currently does not support is cause by omission. In such a case, the *absence of* X is the cause of Y. For example, the failure of Suzy to shoot the turkey is a cause for it being alive at the end of a scenario. In this case, it is easy to imagine that searching paths for transition states would not be a reasonable approach to solve this, rather, we may want to employ some kind of counterfactual

reasoning to determine if there are any possible counterfactual *divergences* from the scenario that would have resulted in the outcome *not* being true. Another example of cause by omission might involve modeling (possibly continuous) time-dependent effects. Say for instance that a grandfather clock's casing is too small for the swinging range of its pendulum. If the clock-maker does not stop the pendulum from swinging, then the casing will eventually break. In this case, searching for a transition state *would* make sense if we want to know why the casing has broken, but some counterfactual reasoning would still be required to reason about what could have happened to prevent this outcome. Cause by omission has been studied extensively in [78] and [79], among others, although the approaches cited here would appear to serve as a good starting point for research as their work is also based on models of causation that are inspired by how humans reason about causal influence. We also speculate that it may be possible to extend our framework to address problems of this type by leveraging recent work in planning for hybrid domains [80, 81].

Finally, the specification of sophisticated, real-world inspired use cases should be explored to identify requirements for the framework's application in practical settings. Exploring these lines of investigation is likely to yield results and intuitions that lead to the discovery of interesting (possibly unexpected) application areas, the development of novel and challenging benchmark datasets, and may eventually inform the design and development of complex, intelligent software systems that are able to explain their behaviors and decisions in intuitive ways.

Bibliography

[1] Charles E Carpenter. Concurrent causation. *University of Pennsylvania Law Review and American Law Register*, 83(8):941–952, 1935.

[2] Judea Pearl. Causality: models, reasoning and inference. *Econometric Theory*, 19(675-685):46, 2000.

[3] Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part i: Causes. *The British journal for the philosophy of science*, 56(4):843–887, 2005.

[4] Joseph Y Halpern. Axiomatizing causal reasoning. *Journal of Artificial Intelligence Research*, 12: 317–337, 2000.

[5] Judea Pearl. On the definition of actual cause. Technical report, University of California, 1998.

[6] Brad Weslake. A partial theory of actual causation. *British Journal for the Philosophy of Science*, 2015.

[7] Ned Hall, Laurie A Paul, et al. Causation and preemption. *Philosophy of science today*, pages 100–130, 2003.

[8] Ned Hall. Structural equations and causation. *Philosophical Studies*, 132(1):109–136, 2007.

[9] Sander Beckers and Joost Vennekens. A general framework for defining and extending actual causation using cp-logic. *International Journal of Approximate Reasoning*, 77:105–126, 2016.

[10] David Lewis. Causation. *The journal of philosophy*, 70(17):556–567, 1973.

[11] Clark Glymour and David Danks. Actual causation: a stone soup essay. *Synthese*, 175(2): 169–192, 2010.

[12] Peter Menzies. Counterfactual theories of causation. *The Stanford Encyclopedia of Philosophy*, 2001.

[13] Mark Hopkins and Judea Pearl. Causality and counterfactuals in the situation calculus. *Journal of Logic and Computation*, 17(5):939–953, 2007.

[14] Joseph Y Halpern. *Actual causality*. MIT Press, 2016.

[15] Sander Beckers and Joost Vennekens. The Transitivity and Asymmetry of Actual Causation. *Ergo, an Open Access Journal of Philosophy*, 4, January 2017. doi: 10.3998/ergo.12405314.0004.001. URL `https://lirias.kuleuven.be/handle/123456789/570017`.

[16] Marc Denecker, Bart Bogaerts, and Joost Vennekens. Explaining actual causation in terms of possible causal processes. In *16th Edition of the European Conference on Logics in Artificial Intelligence*. Springer, 2019.

[17] Joost Vennekens. Actual causation in cp-logic. *Theory and Practice of Logic Programming*, 11 (4-5):647–662, 2011.

[18] Chitta Baral and Michael Gelfond. Reasoning agents in dynamic domains. In *Logic-based artificial intelligence*, pages 257–279. Springer, 2000.

[19] Vitaliy Batusov and Mikhail Soutchanski. Situation calculus semantics for actual causality. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[20] Vitaliy Batusov and Mikhail Soutchanski. Situation calculus semantics for actual causality. In *13th International Symposium on Commonsense Reasoning. University College London, UK. Monday, November*, volume 6, 2017.

[21] Joost Vennekens. Actual causation in cp-logic. *Theory and Practice of Logic Programming*, 11 (4-5):647–662, 2011.

[22] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1-2):49–104, 2004.

[23] Raymond Reiter. Natural actions, concurrency and continuous time in the situation calculus. *KR*, 96:2–13, 1996.

[24] Sandeep Chintabathina, Michael Gelfond, and Richard Watson. Modeling hybrid domains using process description language. In *In Proceedings of ASP '05 Answer Set Programming: Advances in Theory and Implementation,*, volume 6, pages 303–317.

[25] Chitta Baral, Nam Tran, and Le-Chi Tuan. Reasoning about actions in a probabilistic setting. In *AAAI/IAAI*, pages 507–512, 2002.

[26] Chitta Baral, Michael Gelfond, and Nelson Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9(1):57–144, 2009.

[27] Michael Gelfond and Richard Watson. Diagnostics with answer sets: Dealing with unobservable fluents. In *Proceedings of the 3rd International Workshop on Cognitive Robotics-CogRob'02*, pages 44–51, 2002.

[28] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Readings in artificial intelligence*, pages 431–450, 1969.

[29] Vladimir Lifschitz. Formal theories of action (preliminary report). In *IJCAI*, pages 966–972. Citeseer, 1987.

[30] Brian A Haugh. Simple causal minimizations for temporal persistence and projection. In *AAAI*, pages 218–223, 1987.

[31] Michael Gelfond and Vladimir Lifschitz. Representing Action and Change by Logic Programs. *Journal of Logic Programming*, 17(2–4):301–321, 1993.

[32] Fangzhen Lin. Embracing causality in specifying the indeterminate effects of actions. In *Proceedings of the thirteenth national conference on Artificial intelligence-Volume 1*, pages 670–676. AAAI Press, 1996.

[33] Norman McCain, Hudson Turner, et al. Causal theories of action and change. In *AAAI/IAAI*, pages 460–465, 1997.

[34] Michael Thielscher. Ramification and causality. *Artificial intelligence*, 89(1-2):317–364, 1997.

[35] Michael Gelfond and Vladimir Lifschitz. Action languages. In *E;ectronic Transactions on AI*, volume 3(16), 1998.

[36] Marcello Balduccini and Emily LeBlanc. Action-centered information retrieval, March 7 2019. US Patent App. 16/121,674.

[37] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *IJCAI*, volume 95, pages 1978–1984, 1995.

[38] Vladimir Lifschitz. On the logic of causal explanation. *Artificial Intelligence*, 96(2):451–465, 1997.

[39] Patrick J. Hayes and John McCarthy. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[40] Murray Shanahan et al. *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. MIT press, 1997.

[41] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080, 1988.

[42] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New generation computing*, 9(3-4):365–385, 1991.

[43] Marcello Balduccini and Michael Gelfond. Diagnostic reasoning with A-Prolog. *Journal of Theory and Practice of Logic Programming (TPLP)*, 3(4–5):425–461, Jul 2003.

[44] Jürgen Dix, Ugur Kuter, and Dana Nau. Planning in answer set programming using ordered task decomposition. In *Annual Conference on Artificial Intelligence*, pages 490–504. Springer, 2003.

[45] Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. Answer set planning under action costs. *Journal of Artificial Intelligence Research*, 19:25–71, 2003.

[46] Esra Erdem, Michael Gelfond, and Nicola Leone. Applications of answer set programming. *AI Magazine*, 37(3), 2016.

[47] Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Francesco Ricca, and Torsten Schaub. Asp-core-2: Input language format. *ASP Standardization Working Group*, 2012.

[48] Emily LeBlanc, Marcello Balduccini, and Joost Vennekens. Explaining actual causation via reasoning about actions and change. In *European Conference on Logics in Artificial Intelligence*, pages 231–246. Springer, 2019.

[49] Ned Hall. Two concepts of causation. *Causation and counterfactuals*, pages 225–276, 2004.

[50] Sander Beckers and Joost Vennekens. The transitivity and asymmetry of actual causation. 2017.

[51] Bertram F Malle, Steve Guglielmo, and Andrew E Monroe. A theory of blame. *Psychological Inquiry*, 25(2):147–186, 2014.

[52] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial intelligence*, 33(3):379–412, 1987.

[53] Emily LeBlanc. Explaining actual causation via reasoning about actions and change. In *Technical Communications of the 34th Int'l. Conference on Logic Programming (ICLP'18)*, August 2018.

[54] Judea Pearl. Causality: models, reasoning, and inference. 2000.

[55] Kichun Jo, Junsoo Kim, Dongchul Kim, Chulhoon Jang, and Myoungho Sunwoo. Development of autonomous car—part i: Distributed system architecture and development process. *IEEE Transactions on Industrial Electronics*, 61(12):7131–7140, 2014.

[56] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In *ICLP*, volume 94, pages 23–37, 1994.

[57] Hudson Turner. Splitting a default theory. In *AAAI/IAAI, Vol. 1*, pages 645–651, 1996.

[58] David Hume. A treatise of human nature [1739]. *British Moralists*, pages 1650–1800, 1978.

[59] Brad Weslake. A partial theory of actual causation. 2015.

[60] Thomas Eiter and Thomas Lukasiewicz. Complexity results for structure-based causality. *Artif. Intell.*, 142(1):53–89, November 2002. ISSN 0004-3702. doi: 10.1016/S0004-3702(02)00271-0. URL `http://dx.doi.org/10.1016/S0004-3702(02)00271-0`.

[61] Chitta Baral and Matt Hunsaker. Using the probabilistic logic programming language p-log for causal and counterfactual reasoning and non-naive conditioning. In *IJCAI*, pages 243–249, 2007.

[62] Dan B Dobbs. Rethinking actual causation in tort law. 2017.

[63] Glenn Shafer. *The art of causal conjecture*. MIT press, 1996.

[64] Raymond Reiter. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT press, 2001.

[65] Jean-François Bonnefon, Rui Da Silva Neves, Didier Dubois, and Henri Prade. Qualitative and quantitative conditions for the transitivity of perceived causation. *Annals of Mathematics and Artificial Intelligence*, 64(2-3):311–333, 2012.

[66] Emily LeBlanc and Marcello Balduccini. Contextual representations of cause via reasoning about actions and change. 2017.

[67] Raymond Reiter. The situation calculus ontology. Technical report, Electronic News Journal on Reasoning about Actions and Change, 1997.

[68] Fangzhen Lin and Yoav Shoham. Concurrent actions in the situation calculus. In *AAAI*, volume 92, pages 590–595, 1992.

[69] Pedro Cabalar. Causal logic programming. *Correct Reasoning*, 7265:102–116, 2012.

[70] Jorge Fandinno. Towards deriving conclusions from cause-effect relations. *Fundamenta Informaticae*, 147(1):93–131, 2016.

[71] Alexander Bochman and Vladimir Lifschitz. Pearl's causality in a logical setting. In *AAAI*, pages 1446–1452, 2015.

[72] Pedro Cabalar, Jorge Fandinno, and Michael Fink. Causal graph justifications of logic programs. *Theory and Practice of Logic Programming*, 14(4-5):603–618, 2014.

[73] Enrico Pontelli, Tran Cao Son, and Omar Elkhatib. Justifications for logic programs under answer set semantics. *Theory and Practice of Logic Programming*, 9(1):1–56, 2009.

[74] Jorge Fandinno. Deriving conclusions from non-monotonic cause-effect relations. *Theory and Practice of Logic Programming*, 16(5-6):670–687, 2016.

[75] Luıs Moniz Pereira and Ari Saptawijaya. Counterfactuals, logic programming and agent morality. *R. Urbaniak, G. Payette (eds.), Applications of Formal Philosophy: The Road Less Travelled, Springer Logic, Argumentation & Reasoning series*, 20187.

[76] Jean-François Bonnefon, Rui Da Silva Neves, Didier Dubois, and Henri Prade. Predicting causality ascriptions from background knowledge: Model and experimental validation. *International Journal of Approximate Reasoning*, 48(3):752–765, 2008.

[77] Salem Benferhat, Jean-François Bonnefon, Philippe Chassy, Rui Da Silva Neves, Didier Dubois, Florence Dupin de Saint-Cyr, Daniel Kayser, Farid Nouioua, Sara Nouioua-Boutouhami, Henri Prade, et al. A comparative study of six formal models of causal ascription. In *International Conference on Scalable Uncertainty Management*, pages 47–62. Springer, 2008.

[78] Paul Bello and Sangeet Khemlani. A model-based theory of omissive causation. In *CogSci*, 2015.

[79] Sangeet Khemlani, Christina Wasylyshyn, Gordon Briggs, and Paul Bello. Mental models and omissive causation. *Memory & cognition*, 46(8):1344–1359, 2018.

[80] Marcello Balduccini, Daniele Magazzeni, Marco Maratea, and Emily C Leblanc. Casp solutions for planning in hybrid domains. *Theory and Practice of Logic Programming*, 17(4):591–633, 2017.

[81] Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research*, 27:235–297, 2006.

Vita

Emily Cooper LeBlanc received a Bachelor of Science in Computer Science from Temple University in Philadelphia, Pennsylvania in May of 2013. She received her Masters in Computer Science from Drexel University in Philadelphia, Pennsylvania in June of 2017. She received her PhD in Computer Science with a focus on Logic-Based Causal Reasoning. In her time at Drexel, she published two journal articles, one main conference paper, two technical communications, and seven workshop papers. She won the Best Doctoral Consortium Paper award at the 34th International Conference on Logic Programming in 2018. Publication titles include:

- Reasoning about Problems of Actual Causation using an Action Language Approach
- Explaining Actual Causation via Reasoning about Actions and Change
- Action-Centered Information Retrieval
- CASP Solutions for Planning in Hybrid Domains
- Military Ontologies for Information Dissemination at the Tactical Edge
- Ontologies and Rich Metadata for Materials Scientific Data Analysis

The full list of publications can be found on her website at `www.eleblanc.ai`. She has two pending patents. She has been involved in a number of research projects in her time at Drexel, including work with the Defense Advanced Research Projects Agency (DARPA), the National Science Foundation (NSF), and the Federal Highway Administration (FHWA). She was a teaching assistant for approximately two years in total, and was promoted to Lead Graduate Teaching Assistant in 2019. She is a member of the Upsilon Pi Epsilon Honors Society (UPE). She has given multiple invited talks, the most recent of which was given on her dissertation work at the Navy Center for Applied Research in Artificial Intelligence (NCARAI) Symposium Series at the U.S. Naval Research Laboratory in 2019.