# The Theory of Boolean Counterfactual Reasoning

Kilian Rückschloß

München 2024

# The Theory of Boolean Counterfactual Reasoning

**Kilian Rückschloß**

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universiät München

eingereicht von
Kilian Rückschloß
am 04.06.2024

Erstgutachter: Prof. Dr. François Bry
Zweitgutachter: Prof. Dr. Fabrizio Riguzzi
Tag der mündlichen Prüfung: 15.07.2024

## Versicherung an Eides statt

(gemäß § 8 Abs. 2 Nr. 5 der Promotionsordnung vom 12. Juli 2011)

Hiermit erkläre ich, Kilian Rückschloß, an Eides statt, dass die Dissertation mit dem Titel „The Theory of Boolean Counterfactual Reasoning“ von mir selbstständig und ohne unerlaubte Beihilfe angefertigt wurde.

München, den 19.07.2024
Ort, Datum

Unterzeichner

# Zusammenfassung

Ursache und Wirkung bilden eine Grundlage dafür, wie Menschen ihre Umwelt wahrnehmen. Wir erklären unsere Umwelt, indem wir Beobachtungen als Wirkung anderer, selbsterklärender Beobachtungen beschreiben. Angenommen, ein Haus brennt nach einem Blitzschlag. In diesem Fall können wir den Blitz als selbsterklärendes Ereignis betrachten, welches das Feuer in dem Haus verursacht. Künstliche Intelligenz hingegen basiert derzeit größtenteils auf dem Konzept der Korrelation. Dies bedeutet, dass sie ausdrücken kann, wie die Beobachtung eines Feuers die Wahrscheinlichkeit für einen Blitzschlag erhöht und umgekehrt. Eine auf Korrelation basierende künstliche Intelligenz kann jedoch den Blitz nicht als Ursache des Feuers erkennen. Während korrelationsbasierte künstliche Intelligenz nur Fragen zur Wahrheit oder Wahrscheinlichkeit von Aussagen beantworten kann, ermöglichen kausale Erklärungen die Beantwortung zweier weiterer Fragetypen: Fragen zu den Folgen externer Interventionen und kontrafaktische Fragen, also Fragen der Form: „Was wäre, wenn...?“. Man kann beispielsweise intervenieren und einen Blitzableiter installieren, um Brände zu verhindern, die durch Blitzschläge verursacht werden. Beobachten wir einen Blitzschlag, der ein Feuer in einem Haus ohne Blitzableiter verursacht, folgt aus dieser kausalen Erklärung, dass kein Feuer ausgebrochen wäre, hätte man einen Blitzableiter installiert.

Menschen beschreiben ihre Umwelt nicht nur in Form kausaler Ursache-Wirkungs-Beziehungen. Oft sind wir auch mit unsicherem Wissen oder mit Beziehungen zwischen Individuen konfrontiert. Wenn wir beispielsweise eine Gruppe von Menschen betrachten, können zwei Individuen innerhalb dieser Gruppe Freunde sein, d.h., sie stehen in Beziehung zueinander. Darüber hinaus kann man annehmen, dass Freunde von Rauchern mit erhöhter Wahrscheinlichkeit ebenfalls rauchen. Wir sind uns also unsicher, ob Freunde von Rauchern zwangsläufig selbst rauchen. Schließlich nehmen wir an, dass die Wahrscheinlichkeit, Raucher zu sein, für alle Personen mit derselben Anzahl von rauchenden Freunden übereinstimmt. Die Annahme, dass Individuen mit denselben Eigenschaften sich gleich verhalten, kann Berechnungen vereinfachen oder den Raum möglicher Beschreibungen eines Datensatzes verkleinern. Das Feld der statistischen relationalen künstlichen Intelligenz kombiniert die prädikatenlogische Repräsentation von Beziehungen innerhalb eines Bereichs mit der Repräsentation von Unsicherheit durch Wahrscheinlichkeiten. Hierbei dient Prädikatenlogik als Sprache, um die Austauschbarkeit von Individuen in einem Modell oder Datensatz auszudrücken. Dieser Ansatz führte unter anderem zu Markov-Logik-Netzwerken und probabilistischer Logikprogrammierung.

Diese Monographie beginnt mit einem Überblick über die zuvor genannten Formalismen und führt in Pearls Kausalitätstheorie ein. Statt jeden Formalismus einzeln um kausales Schließen zu erweitern, betrachtet diese Arbeit einen axiomatischen Ansatz. Sie erweitert Bochmans Theorie der kausalen Logik zu einer Axiomatisierung des deterministischen kausalen Schließens im Boolschen Fall. Diese Axiomatisierung führt dann zur abduktiven Logikprogrammierung als Sprache für kausales Wissen. Anschließend werden abduktive Logikprogramme mit Markov-Logik-Netzwerken zu gewichteten abduktiven Logikprogrammen kombiniert. Auf diese Weise erhält man einen allgemeinen Formalismus für kausales Schließen mit unsicherem Wissen. Die Arbeit spezifiziert die kausalen Fragentypen, d.h. Fragen zu den Folgen externer Interventionen und kontrafaktische Fragen, im Kontext der gewichteten abduktiven Logikprogrammierung. Durch Einbettung verschiedener Formalismen aus dem Feld der statistisch relationalen künstlichen Intelligenz wird das gegebene kausale Schließen übertragen und konsistentes kausales Schließen über verschiedene Formalismen hinweg garantiert. Weiter wird kontrafaktisches Schließen in azyklischen ProbLog-Programmen untersucht. Ein Resultat gibt Regularitätsbedingungen, unter denen azyklische ProbLog-Programme aus ihrer kontrafaktischen Ausgabe rekonstruiert werden können. ProbLog scheint daher als Sprache für kontrafaktisches Schließen besonders geeignet zu sein. Die Lerntheorie von ProbLog-Programmen basiert jedoch auf statistischen Tests, die keine Informationen über den der Daten zugrunde liegenden kausalen Mechanismus liefern. Somit ist es unzulässig, die resultierenden Programme für kontrafaktisches Schließen zu verwenden. Im letzten Teil wird ein Fragment beschrieben, in welchem sich ProbLog-Programme aus ihrer Verteilung rekonstruieren lassen. Weiter wird argumentiert, dass dieses Fragment einen Ansatz liefert, um ProbLog-Programme für kontrafaktisches Schließen aus Daten zu lernen.

# Abstract

Human reasoning is heavily based on distinguishing causes from their effects. We typically understand our environment by explaining observations, i.e., effects, with the help of self-evident a priori knowledge, i.e., causes. Assume, for instance, that a lightning strike hits a house and the house burns down. In this case, we consider the lightning as self-evident, a priori knowledge, which explains the fire in the house. This leads us to the judgment: “The lightning caused the fire in the house”.

Unfortunately, nowadays artificial intelligence is mostly built on the concept of correlation. In our example, this means we can express that observing a fire increases the probability of a lightning strike hitting our house, and vice versa. However, a correlation-based artificial intelligence cannot recognize the lightning as the fire’s cause. While correlation-based artificial intelligence only supports queries about the truth or probabilities of statements, causal explanations allow two additional query types: queries for the effects of external interventions and counterfactual queries, i.e., queries of the form: “What if...?”. In our scenario, we can, for instance, intervene and install a conductor to prevent fires being caused, i.e. explained by lightning strikes. Furthermore, if we observe a lightning strike hitting our house followed by a fire breaking out, we may conclude from a causal explanation that the fire would not have occurred had we installed a lightning rod beforehand.

Humans reason not only on causes, but also on uncertainties, represented by probabilities, and on relations between components or individuals of a given domain formalized in logic, preferably first-order logic. When, for example, given a group of people, two individuals within this group may be friends, i.e., they share a relationship. Moreover, one may assume that friends of smokers are more likely to be themselves smokers, meaning that we are uncertain about the implication whether friends of smokers necessarily smoke. Finally, we may assume that all individuals with the same number of smoking friends are equally likely to smoke themselves. The assumption that individuals with the same properties behave in the same way may speed up calculations or shrink the space of possible descriptions for a dataset. The field of statistical relational artificial intelligence combines first-order logic reasoning on relations in a domain and probabilistic reasoning on uncertainty, whereby first-order logic serves as a language for expressing the interchangeability of individuals in a probabilistic model or dataset. Among other things, this approach has so far resulted in Markov logic networks and probabilistic logic programs.

The work reported in this monograph first provides a brief overview of the aforementioned formalisms and introduces Pearl’s causal reasoning. Instead of continuously expanding every distinct formalism with causal reasoning, the thesis proposes an axiomatic approach to Boolean causal reasoning under uncertainty. Initially, we extend Bochman’s logical theory of causality to obtain a complete axiomatization of deterministic Boolean causal reasoning, leading to abductive logic programming. We then generalize abductive logic programming to the framework of Markov logic networks, which employs a maximum entropy approach extending first-order logic to handle uncertainty. In this way, we obtain weighted abductive logic programming as a general framework tailored to Boolean causal reasoning under uncertainty. We specify the causal query types, i.e., queries for the effects of external interventions and counterfactual queries, within the context of weighted abductive logic programming. By embedding a formalism into our framework and subsequently transferring the causal reasoning there, we derive causal reasoning in widespread formalisms of statistical relational artificial intelligence. In particular, our approach guarantees consistent causal reasoning across these frameworks. Next, we focus on counterfactual reasoning in acyclic ProbLog programs, where the proposed approach is implemented in Kiesel’s WhatIf solver. We show that sufficiently well-behaved ProbLog programs can be reconstructed from their counterfactual output. Hence, ProbLog is a particularly suitable framework to address counterfactual reasoning. However, learning ProbLog programs from data is usually based on statistical tests, lacking information about the underlying causal mechanism. This makes it infeasible to use the resulting programs for counterfactual reasoning. To address this issue in the last part, we propose a fragment of ProbLog that allows reconstructing a program from its induced distribution, finally enabling us to learn programs supporting counterfactual queries.

# Acknowledgements

First, I am grateful to Dr. Felix Weitkämper for his invaluable advice, continuous support, fruitful discussions, and patience during my doctoral studies. I am also grateful to my supervisor, Prof. Dr. François Bry. His knowledge and extensive experience have encouraged me throughout my academic research and daily life. I thank Prof. Dr. Fabrizio Riguzzi for pointing out a mistake in the introduction of the P-log semantics, which helped me to correct the semantics of weighted abductive causal theories and weighted abductive logic programs later on. I also thank Prof. Dr. Jasmin Blanchette for his valuable comments on my thesis. I would like to thank all my colleagues. Their support and good company have made my doctoral journey a wonderful experience.

I thank Louisa Sigl for helping me with the application for my current position and Kailin Sun for frequently helping me with my English. I thank Prof. Dr. Oliver Deiser for his Analysis lectures in 2011 and 2012. It was his style of explaining mathematics that brought me into this field of research.

I express my deepest gratitude to my parents, Martina Rückschloß-Stadler and Bernhard Rückschloß. Without their understanding and encouragement over the past years, it would have been impossible for me to complete my studies. I am also extremely grateful to my partner, Anna Fleckenstein, for her understanding and support throughout this challenging but rewarding journey, and for the enjoyable moments we shared.

Last but not least, I thank God, the Father, the Son, and the Holy Spirit for creating me and for the energy, health, and inspiration that enabled this work.

# Contents

# Chapter 1

# Introduction

Humans possess the remarkable skill of reasoning in terms of counterfactuals, allowing them to answer questions of the form: "What if...?". This ability proves particularly relevant in fields such as medicine, where we may inquire: "What if a patient in the control group of a pharmaceutical study had received treatment? Would he have recovered?". In decision-making, we might ask: "What if I had taken another flight? Would I still have lost my luggage?". Counterfactual reasoning plays a crucial role in defining the notions of responsibility and blame, as discussed in Halpern's work (30). For example, when considering a driver's responsibility for an accident, we may question: "What if the driver had not sped up? Would the accident still have occurred?". Overall, counterfactuals enable us to make sense of the past, plan future courses of action, make emotional and social judgments, and adapt our behavior (31). Consequently, an artificial intelligence should also be able to reason counterfactually.

Generally, counterfactual reasoning means contemplating how events would have unfolded under different circumstances without actually experiencing these different realities. For example, when we make judgments such as: "I would not have lost my luggage if I had booked a direct flight." we do not experience the reality in which we take a direct flight. Thus, these judgments reflect our ability to rationally derive alternative outcomes based on different choices or actions.

Following Lewis (37), we identify a counterfactual query like: "What if I had taken another flight? Would I still have lost my luggage?" with a triplet consisting of an observation $\mathbf{e}$ (we lost our luggage on the current flight), a constraint $\mathbf{i}$ (we would take another flight) and a query $\phi$ (asking if we still lose our luggage assuming the constraint $\mathbf{i}$ is true). We then aim to answer the query $\phi$ in an alternative world $\omega^a$ that satisfies the constraint $\mathbf{i}$, i.e., where we take another flight, while minimally deviating from the real world $\omega^r$, where we observe the evidence $\mathbf{e}$ that we lose our luggage on the current flight.

Although Lewis (37) appropriately formalizes the concepts underlying counterfactual reasoning, he does not provide a construction for the alternative worlds $\omega^a$. We adapt the idea of Pearl (41) and construct an alternative world by manipulating the causal explanation for the real world $\omega^r$. If we explain the loss of our luggage with a strike at the airport of departure, we are likely to judge that we would also have lost it on another flight. Conversely, if we explain it with a too short transfer time, we may judge that we would not have lost it had we taken another connection. This thesis investigates Pearl's approach to counterfactual reasoning in the Boolean case, where all relevant variables take on two values, represented by true and false.

## 1.1 Counterfactual Reasoning in Pearl's Causal Models

Pearl (41) suggests modeling causal mechanisms with deterministic functional relationships. In his **causal models**, he first distinguishes between explained **internal variables $\mathbf{V}$** and unexplained **external variables $\mathbf{U}$**. The internal variables $V \in \mathbf{V}$ are then explained, i.e., defined by functions $f_V(\mathbf{U}, \mathbf{V})$ in the internal variables $\mathbf{V}$ and the external variables $\mathbf{U}$, such that the system of equations $V := f_V(\mathbf{V}, \mathbf{U})$, $V \in \mathbf{V}$ has a unique solution for all values of the external variables $\mathbf{U}$.

**Example 1.1.1.** Consider a road that passes through a field with a sprinkler in it. Denote by *cloudy* the event that the weather is cloudy. The sprinkler is turned on by a weather sensor if the weather is sunny, that is, if *cloudy* is false. Suppose further that it rains, denoted by rain, whenever it is cloudy, and that the road is wet, denoted by *wet*, if either it rains or the sprinkler is turned on.

According to Pearl (41), this causal mechanism is represented by the causal model $\mathcal{M}$ that consists of the internal variables $\mathbf{V} := \{rain, sprinkler, wet\}$, the external variables $\mathbf{U} := \{cloudy\}$, and the system of equations:

$$sprinkler := \neg cloudy \qquad rain := cloudy \qquad wet := rain \vee sprinkler$$

Given truth values for external variables, that is, $cloudy = True$ or $cloudy = False$, we obtain the solutions:

$$\begin{array}{llll} cloudy = True, & rain = True, & sprinkler = False, & wet = True \\ cloudy = False, & rain = False, & sprinkler = True, & wet = True \end{array}$$

This provides us with the knowledge represented by the solutions, along with the explanation that $cloudy = True$ or $cloudy = False$.

Pearl (41) further observes that his causal models are able to represent external interventions. An external intervention forcing an internal variable $V \in \mathbf{V}$ to attain the value $v$ is represented by replacing the equation $V := f_V(\mathbf{U}, \mathbf{V})$ in $\mathcal{M}$ with $V := v$ resulting in the **modified causal model** or the **submodel** $\mathcal{M}_{V:=v}$.

**Example 1.1.2.** Assume that we manually switch the sprinkler on (or off) in the situation of Example 1.1.1. According to Pearl (41), this modification results in the modified causal model or submodel $\mathcal{M}_{sprinkler:=True}$ ($\mathcal{M}_{sprinkler:=False}$) obtained by replacing the equation for *sprinkler* with $sprinkler := True$ ($False$). Hence, we obtain the modified causal model $\mathcal{M}_{sprinkler:=True}$ ($\mathcal{M}_{sprinkler:=False}$) that consists of the internal variables $\mathbf{V} := \{rain, sprinkler, wet\}$, the external variables $\mathbf{U} := \{cloudy\}$, and the system of equations:

$$\begin{aligned} & sprinkler := True\ (False) \\ & rain := cloudy \\ & wet := rain \vee sprinkler \end{aligned}$$

If we observe the sprinkler to be off in the causal model $\mathcal{M}$ of Example 1.1.1, we can draw the conclusion that the weather is cloudy and it rains. However, if we intervene and turn the sprinkler off manually, we cannot draw such a conclusion in the modified model $\mathcal{M}_{sprinkler:=False}$. Consequently, intervening and switching off the sprinkler is indeed different from observing that the sprinkler is off.

As external interventions often arise from actions, i.e., doing something, such as switching the sprinkler off in Example 1.1.2, Pearl (41) introduces the **do-operator** do(_) to refer to the corresponding modified causal models, i.e., he defines

$$\mathcal{M}_{sprinkler:=True} := \mathcal{M}^{\mathrm{do}(sprinkler:=True)}\ (\mathcal{M}_{sprinkler:=False} := \mathcal{M}^{\mathrm{do}(sprinkler:=False)}).$$

Finally, Pearl (41) argues that an intervention constitutes a minimal change such that a certain constraint becomes true. This viewpoint then allows Balke and Pearl (5) to construct Lewis' alternative worlds (37) to answer counterfactual queries.

**Example 1.1.3.** Recall the scenario of Example 1.1.1 and assume that we observe the sprinkler on and the road wet. What if the sprinkler was turned off? Would the road also have been wet?

To answer this counterfactual query, Balke and Pearl (5) create two copies of the equation system in Example 1.1.1 – one with superscript $a$ to represent the alternative and one with superscript $r$ to represent the real world so that both copies mention the same external variables. This results in the following twin model $\mathcal{M}^T$:

$$\begin{array}{lll} \mathit{sprinkler}^a := \neg \mathit{cloudy} & \mathit{rain}^a := \mathit{cloudy} & \mathit{wet}^a := \mathit{rain}^a \vee \mathit{sprinkler}^a \\ \mathit{sprinkler}^r := \neg \mathit{cloudy} & \mathit{rain}^r := \mathit{cloudy} & \mathit{wet}^r := \mathit{rain}^r \vee \mathit{sprinkler}^r \end{array}$$

To satisfy the constraint on the alternative world, they intervene according to $\mathit{sprinkler}^a := \mathit{False}$ to obtain the modified twin model $\mathcal{M}^T_{\mathit{sprinkler}^a := \mathit{False}}$:

$$\begin{array}{lll} \mathit{sprinkler}^a := \mathit{False} & \mathit{rain}^a := \mathit{cloudy} & \mathit{wet}^a := \mathit{rain}^a \vee \mathit{sprinkler}^a \\ \mathit{sprinkler}^r := \neg \mathit{cloudy} & \mathit{rain}^r := \mathit{cloudy} & \mathit{wet}^r := \mathit{rain}^r \vee \mathit{sprinkler}^r \end{array}$$

Since we observe the sprinkler to be on, indicating $\mathit{sprinkler}^r = \mathit{True}$, we infer that the weather is sunny, i.e., $\mathit{cloudy} = \mathit{False}$. Hence, we can solve the equations and obtain $\mathit{wet}^a = \mathit{False}$. Thus, we conclude that the road would not have been wet if the sprinkler had been turned off.

Unfortunately, in many cases, it is not possible to answer deterministic counterfactual queries.

**Example 1.1.4.** Assume that the sprinkler in Example 1.1.1 may also be turned on due to sensor failure, denoted $\mathit{failure}$. In this case, we obtain the causal model that consists of the internal variables $\mathbf{V} := \{\mathit{rain}, \mathit{sprinkler}, \mathit{wet}\}$, the external variable $\mathbf{U} := \{\mathit{cloudy}, \mathit{failure}\}$, and the system of equations:

$$\mathit{sprinkler} := \neg \mathit{cloudy} \vee \mathit{failure} \qquad \mathit{rain} := \mathit{cloudy} \qquad \mathit{wet} := \mathit{rain} \vee \mathit{sprinkler}$$

If we now ask whether the road would be wet if the sprinkler was off, we proceed as in Example 1.1.3, and obtain the following system of equations:

$$\begin{array}{lll} \mathit{sprinkler}^a := \mathit{False} & \mathit{rain}^a := \mathit{cloudy} & \mathit{wet}^a := \mathit{rain}^a \vee \mathit{sprinkler}^a \\ \mathit{sprinkler}^r := \neg \mathit{cloudy} \vee \mathit{failure} & \mathit{rain}^r := \mathit{cloudy} & \mathit{wet}^r := \mathit{rain}^r \vee \mathit{sprinkler}^r \end{array}$$

If we now explain our observation $\mathit{sprinkler}^r = \mathit{True}$ with $\mathit{cloudy} = \mathit{False}$, we again obtain that $\mathit{wet}^a = \mathit{False}$. Whereas, if we explain $\mathit{sprinkler}^r = \mathit{True}$ with $\mathit{failure} = \mathit{True}$, we could also find $\mathit{wet}^a = \mathit{True}$. Therefore, in this case, it is not possible to answer the counterfactual query in Example 1.1.3.

To avoid counterfactual queries as in Example 1.1.4, this thesis studies probabilistic causal models. Formally, a **probabilistic causal model** consists of a causal model and a joint distribution on the external variables.

**Example 1.1.5.** Let us refine the situation of Example 1.1.1. Assume that the weather is cloudy with probability $\pi_1 := 0.5$, and that clouds cause rain with probability $\pi_2 := 0.6$. Furthermore, assume that the sensor turns on the sprinkler with probability $\pi_3 := 0.1$ if it is cloudy and with probability $\pi_4 := 0.7$ if it is sunny. Finally, assume that rain causes the road to be wet with probability $\pi_5 := 0.8$, and the sprinkler causes the road to be wet with probability $\pi_6 := 0.3$.

We assume that uncertainty in this scenario arises from hidden variables not explicitly modeled. However, their influence is encapsulated in external variables $\mathbf{U} := \{u_1, ..., u_6\}$. For example, the variable $u_3$ represents the possible causes for the sprinkler to be on if it is cloudy. These potential causes, such as sensor failure or children playing and manually switching on the sprinkler, are not explicitly modeled. However, we represent the potential influence of these missing parameters with the external variable $u_3$ and a joint probability distribution $\pi$ on $\mathbf{U}$.

We assume that the external variables $u_1, ..., u_6$ are mutually independent Boolean random variables, with $\pi(u_i) := \pi_i$. The causal relationships are then modeled with the probabilistic causal model $\mathbb{M}$, consisting of the internal variables $\mathbf{V} := \{cloudy, rain, sprinkler, wet\}$, the external variables $\mathbf{U} := \{u_1, ..., u_6\}$, the distribution $\pi$ on $\mathbf{U}$, and the system of equations:

$$\begin{aligned}
&cloudy := u_1\\
&rain := cloudy \wedge u_2\\
&sprinkler := (cloudy \wedge u_3) \vee (\neg cloudy \wedge u_4)\\
&wet := (rain \wedge u_5) \vee (sprinkler \wedge u_6)
\end{aligned}$$

These equations yield a unique solution for each assignment of true values to the external variables $\mathbf{U} := \{u_1, ..., u_6\}$. Thus, the distribution $\pi$ on $\mathbf{U}$ extends to a distribution on $\mathbf{U} \cup \mathbf{V}$. Specifically, rain occurs with probability $\pi(rain) := 0.5 \times 0.6 = 0.3$.

Now, suppose we observe that the road is wet and the sprinkler is on. We want to find the probability $\pi(wet|sprinkler, wet, \mathrm{do}(\neg sprinkler))$ that the road would have been wet if the sprinkler had been off. To answer this counterfactual query, we proceed as in Example 1.1.3 and obtain the following description for the real and alternative worlds, respectively:

$$\begin{aligned}
&cloudy^a := u_1\\
&rain^a := cloudy^a \wedge u_2\\
&sprinkler^a := False\\
&wet^a := (rain^a \wedge u_5) \vee (sprinkler^a \wedge u_6)\\
\\
&cloudy^r := u_1\\
&rain^r := cloudy^r \wedge u_2\\
&sprinkler^r := (cloudy^r \wedge u_3) \vee (\neg cloudy^r \wedge u_4)\\
&wet^r := (rain^r \wedge u_5) \vee (sprinkler^r \wedge u_6).
\end{aligned}$$

Using the resulting probabilistic causal model, that is the modified twin model $\mathbb{M}^T_{sprinkler^a:=False}$, we determine that the road would have been wet if the sprinkler had been off with probability

$$\pi(wet|sprinkler, wet, \mathrm{do}(\neg sprinkler)) := \pi(wet^a|sprinkler^r, wet^r) = 0.18.$$

In particular, we use our observation in the real world to learn about the world outside our model, represented by the external variables $u_1, ..., u_6$ – this is represented by conditioning on the event $\{sprinkler^r, wet^r\}$. Assuming that the real and the alternative world intersect in all influences not explicitly mentioned in our model, we use updated beliefs, i.e. the conditional probabilities of the external variables $u_1, .., u_6$ to answer the query.

$$\begin{aligned}
&\pi(wet|sprinkler, wet, \mathrm{do}(\neg sprinkler)) := \pi(wet^a|sprinkler^r, wet^r) =\\
&= \pi(rain^a \wedge u_5|sprinkler^r, wet^r) = \pi(cloudy^a \wedge u_2 \wedge u_5|sprinkler^r, wet^r) =\\
&= \pi(u_1 \wedge u_2 \wedge u_5|sprinkler^r, wet^r) = 0.18.
\end{aligned}$$

For assignments of values $\mathbf{q}$, $\mathbf{e}$, and $\mathbf{i}$ to internal variables of a probabilistic causal model $\mathbb{M}$, the probability $\pi(\mathbf{q}|\mathbf{e}, \mathrm{do}(\mathbf{i}))$ indicates our confidence level regarding the counterfactual inquiry that $\mathbf{q}$ would have occurred, if $\mathbf{i}$ had taken place before observing the evidence $\mathbf{e}$. Hence, we term $\pi(\mathbf{q}|\mathbf{e}, \mathrm{do}(\mathbf{i}))$ a **counterfactual probability**. This notation, introduced by Papantonis and Belle (40), emphasizes that counterfactual probabilities involve the combination of conditional probabilities and Pearl's interventions, represented by the do-operator $\mathrm{do}(\_)$.

## 1.2 Counterfactual Reasoning in ProbLog

We rely on propositional logic programming to specify causal reasoning for two reasons: The formalism is intuitive and provides a convenient run-time system for computation. Representing our variables with propositions in a propositional alphabet $\mathfrak{P}$, a **logic program $\mathbf{P}$** consists of **clauses** of the form

$$p \leftarrow b_1 \wedge \ldots \wedge b_n$$

for a proposition $p$ and **literals** $b_1, \ldots, b_n$, where $b_i \in \{p_i, \neg p_i\}$ for a proposition $p_i \in \mathfrak{P}$.

**Example 1.2.1.** Recall the situation in Example 1.1.1, i.e. we consider a road that passes through a field with a sprinkler on it. We denote by *cloudy* that the weather is cloudy. A weather sensor turns on the sprinkler if the weather is sunny, i.e., if $\neg cloudy$ holds true. In addition, it rains, denoted by *rain*, whenever it is cloudy and the road is wet, denoted by *wet*, if either it rains or the sprinkler is switched on.

Here, we represent these causal relationships using the following logic program $\mathbf{P}$ in the alphabet $\mathfrak{P} := \{cloudy, sprinkler, rain, wet\}$.

$$\begin{aligned}
&sprinkler \leftarrow \neg cloudy\\
&rain \leftarrow cloudy\\
&wet \leftarrow rain\\
&wet \leftarrow sprinkler
\end{aligned}$$

In a logic program $\mathbf{P}$, we consider a proposition $p \in \mathfrak{P}$ true if the program $\mathbf{P}$ forces $p$ to hold. Otherwise, we apply the concept of **default negation**, stating that the proposition $p$ should be false if it cannot be enforced by the program $\mathbf{P}$. In particular, a negative literal $\neg p$ is true if there is no reason for the proposition $p$ to hold. If there are no cyclic causal relationships at play, this behavior is described by assigning to a logic program $\mathbf{P}$ its **supported models**, i.e., the models $\omega$ of the **Clark completion** (25):

$$\text{comp}(\mathbf{P}) := \left\{ p \leftrightarrow \bigvee_{p \leftarrow b_1 \wedge \ldots \wedge b_n \in \mathbf{P}} b_1 \wedge \ldots \wedge b_n \right\}_{p \in \mathfrak{P}}$$

Here, a model of the Clark completion $\omega$ is a truth value assignment on the propositions $\mathfrak{P}$ that satisfies each formula in comp($\mathbf{P}$). We identify such a truth assignment $\omega$ with the set of all propositions $\omega \subseteq \mathfrak{P}$ that are set to true.

Informally, the Clark completion asserts that every proposition $p$ satisfied in a supported model $\omega$ of a logic program $\mathbf{P}$ must be enforced or supported by at least one clause in $\mathbf{P}$.

**Example 1.2.2.** The Clark completion (25) comp($\mathbf{P}$) of the program $\mathbf{P}$ in Example 1.2.1 is given by the following propositional formulas:

$$cloudy \leftrightarrow \bot \tag{1.1}$$

$$sprinkler \leftrightarrow \neg cloudy \qquad rain \leftrightarrow cloudy \qquad wet \leftrightarrow rain \vee sprinkler \tag{1.2}$$

Hence, we obtain a unique supported model $\omega := \{sprinkler, wet\}$, that is, the sprinkler is on and the road is wet, while the weather is sunny and it does not rain. In particular, we observe that the formulas in Line (1.2) mean the same as the system of equations in Example 1.1.1:

$$\begin{aligned}
&sprinkler := \neg cloudy\\
&rain := cloudy\\
&wet := rain \vee sprinkler
\end{aligned}$$

From Example 1.2.2, we conclude that we still need to find a representation for the external variables of a causal model in logic programming. Hence, we additionally specify a subset of propositions $\mathfrak{A} \subseteq \mathfrak{P}$, the so-called **abducibles**. An **abductive logic program $\mathcal{P}$** is then a pair $\mathcal{P} := (\mathbf{P}, \mathfrak{A})$ consisting of a logic program $\mathbf{P}$ and a set of abducibles $\mathfrak{A} \subseteq \mathfrak{P}$ such that $\mathbf{P}$ contains no clause $p \leftarrow b_1 \wedge \ldots \wedge b_n$ with $p \in \mathfrak{A}$ (22).

**Example 1.2.3.** We represent the causal model of Example 1.1.1 that consists of the internal variables $\mathbf{V} := \{rain, sprinkler, wet\}$, the external variables $\mathbf{U} := \{cloudy\}$ and the system of equations

$$sprinkler := \neg cloudy \qquad rain := cloudy \qquad wet := rain \vee sprinkler$$

by the abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{A})$, where $\mathbf{P}$ is the logic program of Example 1.2.1 and the abducibles are given by $\mathfrak{A} := \mathbf{U} = \{cloudy\}$.

As in causal models, the objective of abductive logic programs is to provide an explanation for a given set of observations. A **(supported) model** $\omega$ of an abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{A})$ with explanation $\epsilon \subseteq \mathfrak{A}$ is a (supported) model of the logic program $\mathbf{P} \cup (\omega \cap \mathfrak{A})$ such that $\epsilon := \omega \cap \mathfrak{A}$.

**Example 1.2.4.** In the case of the abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{A})$ in Example 1.2.3, we obtain the models $\omega_1 := \{sprinkler, wet\}$ with explanation $\epsilon_1 = \emptyset$ and $\omega_2 := \{cloudy, rain, wet\}$ with explanation $\epsilon_2 := \{cloudy\}$. Finally, we note that $\omega_1$ and $\omega_2$ correspond to the solutions

$$\begin{array}{llll} cloudy = True & rain = True & sprinkler = False & wet = True \\ cloudy = False & rain = False & sprinkler = True & wet = True. \end{array}$$

of the causal model in Example 1.2.3.

As Example 1.2.4 illustrates, abductive logic programs are a language for Boolean causal models as long as no cyclic causal relationships are at play. In particular, we can use abductive logic programs to predict the effect of external interventions. Like Vennekens et al. (60) and Riguzzi et al. (48) proposed earlier in probabilistic logic programming, this can be achieved by removing all clauses defining a proposition intervened on and by adding a fact for all propositions that are enforced to be true.

**Example 1.2.5.** Recall the situation described by the abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{A})$ in Example 1.2.4. If we manually turn the sprinkler on, this results in the modified abductive logic program $\mathcal{P}_{sprinkler} := (\mathbf{P}_{sprinkler}, \mathfrak{A})$ obtained by exchanging the logic program $\mathbf{P}$ with the following program $\mathbf{P}_{sprinkler}$.

$$sprinkler \qquad rain \leftarrow cloudy \qquad wet \leftarrow rain \qquad wet \leftarrow sprinkler$$

If we decide to switch the sprinkler manually off, this results in the modified abductive logic program $\mathcal{P}_{\neg sprinkler} := (\mathbf{P}_{\neg sprinkler}, \mathfrak{A})$ obtained by exchanging the logic program $\mathbf{P}$ with the following program $\mathbf{P}_{\neg sprinkler}$.

$$rain \leftarrow cloudy \qquad wet \leftarrow rain \qquad wet \leftarrow sprinkler$$

Following Pearl (41), we argue that an intervention constitutes a minimal change such that a certain constraint becomes true. This point of view then allows us to apply the twin network method of Balke and Pearl (5) for counterfactual reasoning in abductive logic programming.

**Example 1.2.6.** Recall the situation of the abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{A})$ in Example 1.2.3. Once more, assume that we observe the sprinkler to be on and the road to be wet. What if the sprinkler was turned off? Would the road also have been wet?

To answer this counterfactual query, we build the twin program $\mathcal{P}^T := (\mathbf{P}^T, \mathfrak{A})$ and create two copies $\mathbf{P}^{a/r}$ of the logic program $\mathbf{P}$ such that both copies mention the same abducibles, i.e. $cloudy^a = cloudy = cloudy^r$, resulting in the following program $\mathbf{P}^T$.

$$\begin{array}{llll} sprinkler^a \leftarrow \neg cloudy & rain^a \leftarrow cloudy & wet^a \leftarrow rain^a & wet^a \leftarrow sprinkler^a \\ sprinkler^r \leftarrow cloudy & rain^r \leftarrow cloudy & wet^r \leftarrow rain^r & wet^r \leftarrow sprinkler^r \end{array}$$

Again, the superscript $a$ refers to the alternative and the superscript $r$ refers to the real world. To satisfy the constraint on the alternative world, we intervene according to $\neg sprinkler^a$.

$$\begin{array}{llll} & rain^a \leftarrow cloudy & wet^a \leftarrow rain^a & wet^a \leftarrow sprinkler^a \\ sprinkler^r \leftarrow \neg cloudy & rain^r \leftarrow cloudy & wet^r \leftarrow rain^r & wet^r \leftarrow sprinkler^r \end{array}$$

Recall that we observe the sprinkler to be on, i.e, $sprinkler^r$, which is only possible if the weather is sunny, i.e., $\neg cloudy$ holds. Therefore, the only supported model of the modified twin program $\mathcal{P}^T_{\neg sprinkler^a} := (\mathbf{P}^T_{\neg sprinkler^a}, \mathfrak{A})$ that is consistent with our observation is given by

$$\omega := \{sprinkler^r, wet^r\}.$$

In particular, the alternative world is given by

$$\omega^a := \omega \cap \mathfrak{P}^a = \emptyset.$$

As in Example 1.1.3, we find that the road would not have been wet had the sprinkler been off.

Extending abductive logic programs with a distribution that interprets abducibles as mutually independent Boolean random variables leads to the ProbLog language (21; 26).

**Example 1.2.7.** Recall the situation of Example 1.1.5, where the weather is cloudy with probability $\pi_1 := 0.5$ and clouds cause rain with probability $\pi_2 := 0.6$. A sensor turns on the sprinkler with probability $\pi_3 := 0.1$ if it is cloudy and with probability $\pi_4 := 0.7$ if it is sunny. Finally, rain causes the road to be wet with probability $\pi_5 := 0.8$, and the sprinkler causes the road to be wet with probability $\pi_6 := 0.3$.

Again, we model the uncertainty by introducing mutually independent random variables $u_1, ..., u_6$ and distributing them according to $\pi(u_i) := \pi_i$. This results in the following ProbLog program $\mathbb{P}$.

$$0.5 :: u_1 \qquad 0.6 :: u_2 \qquad 0.1 :: u_3 \qquad 0.7 :: u_4 \qquad 0.8 :: u_5 \qquad 0.3 :: u_6$$

$$\begin{array}{l} cloudy \leftarrow u_1 \\ rain \leftarrow cloudy \wedge u_2 \\ sprinkler \leftarrow cloudy \wedge u_3 \\ sprinkler \leftarrow \neg cloudy \wedge u_4 \\ wet \leftarrow rain \wedge u_5 \\ wet \leftarrow sprinkler \wedge u_3 \end{array}$$

Here, the expression $\pi_i :: u_i$ means that we independently choose the facts $u_i$ to be true with probability $\pi_i$. Identifying the logic program with its Clark completion, program $\mathbb{P}$ yields the same distribution as the probabilistic causal model in Example 1.1.5.

Once again, assume that we observe the road to be wet and the sprinkler to be on. In addition, we ask for the probability $\pi(wet|sprinkler, wet, \mathrm{do}(\neg sprinkler))$ that the road would have been wet if the sprinkler had been off. To answer this counterfactual query, we proceed as in Example 1.2.6 and build the following twin program $\mathbb{P}^T$, where the superscripts $a/r$ stand for the real and alternative world, respectively.

$$0.5 :: u_1 \qquad 0.6 :: u_2 \qquad 0.1 :: u_3 \qquad 0.7 :: u_4 \qquad 0.8 :: u_5 \qquad 0.3 :: u_6$$

$$\begin{array}{ll} cloudy^a \leftarrow u_1 & cloudy^r \leftarrow u_1 \\ rain^a \leftarrow cloudy^a \wedge u_2 & rain^r \leftarrow cloudy^r \wedge u_2 \\ sprinkler^a \leftarrow cloudy^a \wedge u_3 & sprinkler^r \leftarrow cloudy^r \wedge u_3 \\ sprinkler^a \leftarrow \neg cloudy^a \wedge u_4 & sprinkler^r \leftarrow \neg cloudy^r \\ wet^a \leftarrow rain^a \wedge u_5 & wet^r \leftarrow rain^r \wedge u_5 \\ wet^a \leftarrow sprinkler^a \wedge u_3 & wet^r \leftarrow sprinkler^r \wedge u_3 \end{array}$$

We intervene according to $\neg sprinkler^a$ and obtain the modified twin program $\mathbb{P}^T_{\neg sprinkler^a}$:

$$0.5 :: u_1 \qquad 0.6 :: u_2 \qquad 0.1 :: u_3 \qquad 0.7 :: u_4 \qquad 0.8 :: u_5 \qquad 0.3 :: u_6$$

$$\begin{array}{ll}
cloudy^a \leftarrow u_1 & cloudy^r \leftarrow u_1 \\
rain^a \leftarrow cloudy^a \wedge u_2 & rain^r \leftarrow cloudy^r \wedge u_2 \\
 & sprinkler^r \leftarrow cloudy^r \wedge u_3 \\
 & sprinkler^r \leftarrow \neg cloudy^r \\
wet^a \leftarrow rain^a \wedge u_5 & wet^r \leftarrow rain^r \wedge u_5 \\
wet^a \leftarrow sprinkler^a \wedge u_3 & wet^r \leftarrow sprinkler^r \wedge u_3
\end{array}$$

Finally, we can use the modified twin program $\mathbb{P}^T_{\neg sprinkler^a}$ to compute that the road would have been wet if the sprinkler had been off with probability

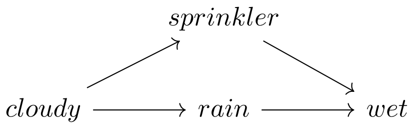
$$\pi(wet|sprinkler, wet, \mathrm{do}(\neg sprinkler)) := \pi(wet^a|sprinkler^r, wet^r) = 0.18.$$

In Section 3.4.3, we present the material of Kiesel et al. (34), where we employ the twin network method of Balke and Pearl (5) to compute counterfactual probabilities in acyclic ProbLog programs as illustrated in Example 1.2.7. Notably, adapting the twin network method of Balke and Pearl (5), Kiesel's WhatIf solver extends his aspmc solver to calculate counterfactual probabilities.

In Chapter 4, we provide a fragment of acyclic ProbLog programs, in which the responses to counterfactual queries uniquely characterize each ProbLog program $\mathbb{P}$ up to syntactic equality. This leads us to the conclusion that ProbLog serves as an effective framework for counterfactual reasoning.

In Chapter 5, we demonstrate that all negation-free ProbLog programs within the fragment introduced in Chapter 4 are uniquely determined up to syntactic equality by their induced distribution and the corresponding cause-effect relationships. These cause-effect relationships can be represented using directed graphs on $\mathfrak{P}$, where an edge $p \to q$ signifies that $p$ is a direct cause of $q$.

**Example 1.2.8.** The case-effect relationship in Example 1.2.7 is given by the following directed graph.



In summary, our findings demonstrate that the probabilities $\pi(\phi) = \pi(\phi|\emptyset, \mathrm{do}(\emptyset))$ and the corresponding cause-effect relationships govern the behavior of these programs entirely. Suppose that we are provided with data sampled from a negation-free ProbLog program $\tilde{\mathbb{P}}$ within the fragment outlined in Chapter 4. Any structure learning algorithm capable of deducing a program within this fragment, along with the correct cause-effect relationships and distribution, reconstructs $\tilde{\mathbb{P}}$ from the data with precision up to syntactic equality. Importantly, the result of this structure learning algorithm is accurate, that is, it provides the same answers to every counterfactual query as the hidden program $\tilde{\mathbb{P}}$.

## 1.3 Bochman's Logical Theory of Causality

Representing causal mechanisms with functional relationships, Sections 1.1 and 1.2 argue that Boolean causal models and abductive logic programs under the supported model semantics emerge as the formalisms of choice for causal modeling when no cyclic causal relationships are at play. Next, we introduce Bochman's causal theories (10) to investigate this modeling assumption.

Bochman (10) takes a more general, logic-based approach to modeling causal mechanisms. Following Aristotle's *Posterior Analytics* (4), he distinguishes between **factual knowledge**, i.e. observations or knowledge about what is, and *scientific knowledge*, which is factual knowledge explained by causal reasoning. In this thesis, we use the term **Aristotelian knowledge** to refer to this notion of scientific knowledge.

**Example 1.3.1.** If we observe a house fire, this constitutes factual knowledge. Suppose further that a police inspector finds that the fire is caused by arson. In this case, the police inspector possesses Aristotelian knowledge, as he not only knows that the house is burning but also understands the reason why it is burning, i.e., arson.

Bochman (10) then aims for a logical theory of causal reasoning, whose semantics captures the concept of Aristotelian knowledge. Formally, he models factual knowledge using propositional logic, i.e. propositional formulas. He then considers causal reasoning as a binary relation $(\Rightarrow)/2$ on factual knowledge that satisfies well-motivated axioms, where $\phi \Rightarrow \psi$ indicates that a causal explanation of the propositional formula $\phi$ also explains the propositional formula $\psi$.

**Example 1.3.2.** Recall the situation in Example 1.1.1 and 1.2.1, i.e., we consider a road that passes along a field with a sprinkler in it and choose a propositional alphabet

$$\mathfrak{P} := \{cloudy, sprinkler, rain, wet\}.$$

Here, *cloudy* means that the weather is cloudy and the proposition *sprinkler* indicates that the sprinkler is on. A weather sensor turns the sprinkler on if the weather is sunny, i.e. if $\neg cloudy$ holds. Rainy weather, indicated by *rain*, occurs whenever it is cloudy and the road is wet, indicated by *wet*, if either it rains or the sprinkler is turned on.

The statement $wet \rightarrow sprinkler \vee rain$ means that once we observe the road to be wet, the sprinkler must be on or it must rain. However, we cannot use the wet road to explain why the sprinkler is on or why it rains, i.e. we only gain factual knowledge about $rain \vee sprinkler$. Whereas $sprinkler \Rightarrow wet$ means both that the road is wet if the sprinkler is on and that the event of the sprinkler being on causally explains why the road is wet, i.e. we gain Aristotelian knowledge about *wet* that is explained by *sprinkler*.

Humans, however, do not state causal knowledge by explicating the entire binary relation of cause and effect; we usually state a few causal rules of the form $\phi \Rightarrow \psi$ and infer the rest of our causal knowledge from these rules. Bochman (10) also represents causal reasoning with **causal theories** $\Delta$, i.e. sets of **causal rules** $\phi \Rightarrow \psi$ for two propositional formulas $\phi$ and $\psi$. In this case, we obtain the result that a formula $\rho$ causally explains a formula $\sigma$ if $\rho \Rightarrow \sigma$ is derivable from $\Delta$ and Bochman's axioms.

**Example 1.3.3.** Intuitively, the situation in Example 1.3.2 is captured in the following causal theory.

$$\begin{aligned} &\neg cloudy \Rightarrow sprinkler \\ &cloudy \Rightarrow rain \\ &sprinkler \Rightarrow wet \\ &rain \Rightarrow wet \end{aligned}$$

However, note that this causal theory cannot explain the proposition *cloudy* as there is no rule of the form ... $\Rightarrow cloudy$.

External variables in causal models and abducibles in abductive logic programs represent parameters that we do not wish to explain. In Bochman's framework, this is indicated by stating that a proposition $p \in \mathfrak{P}$ is self-explained, formally denoted by the **default rule** $p \Rightarrow p$.

**Example 1.3.4.** In Example 1.3.2, we do not expect an explanation for *cloudy*. Formally, this means that *cloudy* is self-explained, that is, we have $cloudy \Rightarrow cloudy$. Together with Example 1.3.3, this yields the following causal theory.

$$\begin{aligned} &cloudy \Rightarrow cloudy \\ &\neg cloudy \Rightarrow sprinkler \\ &cloudy \Rightarrow rain \\ &sprinkler \Rightarrow wet \\ &rain \Rightarrow wet \end{aligned}$$

Finally, we observe that it is impossible to explain the event of a dry road, i.e., $\neg wet$, with the causal theory above as there is no causal rule of the form ... $\Rightarrow \neg wet$.

To solve the problem described in Example 1.3.4, we again employ the concept of default negation. This means we assume that all parameters in our model have a default value that does not require further explanation. Since all parameters are evaluated as true or false, we can further choose our alphabet $\mathfrak{P}$ so that all parameters are false by default. This aligns with intuition in many real-world examples: For example, a schedule indicates when trains or flights depart and not when nothing is departing. Hence, if not explicitly stated, we assume by default that no flight or train leaves. Similarly, we usually assume that houses do not burn. Hence, if we have no reason to believe in a house fire, we assume that houses do not burn.

Specifically, default negation means that we accept having no explanations for a proposition $p$ as an explanation for its negation $\neg p$. In Bochman's framework, this assertion is stated by asserting the default rules $\neg p \Rightarrow \neg p$ for every proposition $p$.

**Example 1.3.5.** Additionally, assuming that all propositions are false by default in the situation of Example 1.3.4, means to form the following negative completion $\Delta^{nc}$ of the causal theory $\Delta$ in Example 1.3.4.

$$\begin{array}{ll} cloudy \Rightarrow cloudy & \neg cloudy \Rightarrow \neg cloudy \\ \neg cloudy \Rightarrow sprinkler & \neg sprinkler \Rightarrow \neg sprinkler \\ cloudy \Rightarrow rain & \neg rain \Rightarrow \neg rain \\ sprinkler \Rightarrow wet & \neg wet \Rightarrow \neg wet \\ rain \Rightarrow wet & \end{array}$$

In this thesis, we are not interested in arbitrary causal theories. Instead, we focus on those causal theories that represent abductive logic programs. Let $\mathcal{P} := (\mathbf{P}, \mathfrak{A})$ be an abductive logic program. We identify $\mathcal{P}$ with the causal theory $\Delta(\mathcal{P})$ that consists of a causal rule $b_1 \wedge \ldots \wedge b_n \Rightarrow p$ for every clause $p \leftarrow b_1 \wedge \ldots \wedge b_n$ in $\mathbf{P}$ and the default rules $u \Rightarrow u$ for the abducibles $u \in \mathfrak{A}$. To reflect the principle of default negation, we further form the **negative completion** $\Delta(\mathcal{P})^{nc}$ and add the default rules $\neg p \Rightarrow \neg p$ for all propositions $p \in \mathfrak{P}$.

**Example 1.3.6.** Recall the abductive logic program $\mathcal{P} = (\mathbf{P}, \mathfrak{A})$ in Example 1.2.3 that consists of the logic program

$$\begin{aligned} &sprinkler \leftarrow \neg cloudy \\ &rain \leftarrow cloudy \\ &wet \leftarrow rain \\ &wet \leftarrow sprinkler \end{aligned}$$

and the abducibles $\mathfrak{A} := \{cloudy\}$. In this case, the causal theory $\Delta(\mathcal{P})$ is given in Example 1.3.4. Furthermore, Example 1.3.5 presents the negative completion $\Delta(\mathcal{P})^{nc}$ of $\Delta(\mathcal{P})$.

Applying Leibniz's principle of sufficient causation (36, p. 268), Bochman asserts that every observation needs to be explained by causal reasoning. Given the negative completion $\Delta(\mathcal{P})^{nc}$ of the causal theory $\Delta(\mathcal{P})$ associated with an abductive logic program $\mathcal{P}$, this means that every proposition $p$ that holds in an observed world $\omega$ needs to be explained by at least one causal rule, i.e., clause in $\mathbf{P}$. Hence, it is only possible to observe worlds $\omega$ that are models of the following completion (10, Theorem 8.115):

$$\operatorname{comp}(\Delta(\mathcal{P})^{nc}) := \left\{ p \leftrightarrow \bigvee_{b_1 \wedge \ldots \wedge b_n \Rightarrow p \in \Delta(\mathcal{P})^{nc}} b_1 \wedge \ldots \wedge b_n \right\}_{p \in \mathfrak{P}} \tag{1.3}$$

Bochman (10) calls the models of $\operatorname{comp}(\Delta(\mathcal{P})^{nc})$ the **causal worlds** of the causal theory $\Delta(\mathcal{P})^{nc}$. Informally, these causal worlds give exactly the (possible) Aristotelian knowledge corresponding to the causal theory $\Delta(\mathcal{P})^{nc}$. Notably, these causal worlds of $\Delta(\mathcal{P})^{nc}$ are exactly the supported models of the abductive logic program $\mathcal{P}$.

**Example 1.3.7.** Recall the abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{A})$ in Example 1.3.6 and the causal theory $\Delta(\mathcal{P})^{nc}$ in Example 1.3.5. According to (1.3), the causal worlds of $\Delta(\mathcal{P})^{nc}$ are exactly the models of the following constraints:

$$\begin{aligned} &\neg cloudy \leftrightarrow sprinkler \\ &cloudy \leftrightarrow rain \\ &sprinkler \vee rain \leftrightarrow wet \end{aligned}$$

Therefore, we obtain the causal worlds:

$$\begin{aligned} &\omega_1 := \{cloudy, rain, wet\} \\ &\omega_2 := \{sprinkler, wet\}. \end{aligned}$$

We observe that these are exactly the supported models of the program $\mathcal{P}$ in Example 1.2.4 and the solutions of the corresponding causal model in Example 1.1.1.

## 1.4 Causal Reasoning in Presence of Cyclic Causal Relationships

In Section 1.3, we introduce Bochman's causal theories (10) as a language for causal reasoning and Aristotelian knowledge. Notably, the reasoning there aligns with the causal reasoning represented in Pearl's causal models (41) and with abductive logic programming (22) under the supported model semantics. Unfortunately, Bochman's causal theories yield counterintuitive results once cyclic causal relationships are at play.

**Example 1.4.1.** Let $h_1$ and $h_2$ be two neighboring houses. Both houses may start to burn, denoted $start\_fire(h_1)$ and $start\_fire(h_2)$, causing a fire in $h_1$ and $h_2$, respectively. Furthermore, assume $h_1$ catches fire, denoted $fire(h_1)$, if $h_2$ burns, denoted $fire(h_2)$, and vice versa. Accepting $start\_fire(h_1)$ and $start\_fire(h_2)$ as self-evident events, this situation is modeled by the abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{A})$, which is given by the logic program

$$\begin{aligned} &fire(h_2) \leftarrow fire(h_1) && fire(h_1) \leftarrow fire(h_2) \\ &fire(h_1) \leftarrow start\_fire(h_1) && fire(h_2) \leftarrow start\_fire(h_2) \end{aligned}$$

and the abducibles $\mathfrak{A} := \{start\_fire(h_1), start\_fire(h_2)\}$.

The causal theory $\Delta(\mathcal{P})^{nc}$ is then given by:

$$\begin{array}{ll}
fire(h_2) \Rightarrow fire(h_1) & fire(h_1) \Rightarrow fire(h_2) \\
start\_fire(h_1) \Rightarrow fire(h_1) & start\_fire(h_2) \Rightarrow fire(h_2) \\
start\_fire(h_1) \Rightarrow start\_fire(h_1) & start\_fire(h_2) \Rightarrow start\_fire(h_2) \\
\neg fire(h_1) \Rightarrow \neg fire(h_1) & \neg fire(h_2) \Rightarrow \neg fire(h_2) \\
\neg start\_fire(h_1) \Rightarrow \neg start\_fire(h_1) & \neg start\_fire(h_2) \Rightarrow \neg start\_fire(h_2)
\end{array}$$

According to (1.3), this results in the following supported models or causal worlds:

$$\begin{aligned}
&\omega_1 := \emptyset \\
&\omega_2 := \{fire(h_1), fire(h_2)\} \\
&\omega_3 := \{start\_fire(h_1), fire(h_1), fire(h_2)\} \\
&\omega_4 := \{start\_fire(h_2), fire(h_1), fire(h_2)\} \\
&\omega_5 := \{start\_fire(h_1), start\_fire(h_2), fire(h_1), fire(h_2)\}
\end{aligned}$$

In the causal world $\omega_2$, both houses $h_1$ and $h_2$ catch fire even though neither of these houses starts burning. This contradicts our everyday causal reasoning, as we do not expect houses to catch fire just because they potentially influence each other.

Example 1.4.1 illustrates that the supported model semantics for abductive logic programming is the wrong choice to model causal mechanisms with cyclic causal relationships. In particular, we obtain the present flaw since Bochman (10) accepts cyclic explanations in his causal theories.

**Example 1.4.2.** The explanation of $fire(h_1)$ in the causal world $\omega_2$ of Example 1.4.1 is given by

$$fire(h_1) \Rightarrow fire(h_2) \text{ and } fire(h_2) \Rightarrow fire(h_1),$$

i.e., we argue that house $h_1$ catches fire because house $h_2$ burns, which catches fire because house $h_1$ burns. In our everyday causal reasoning, however, we do not accept this cyclic argument.

In Section 3.1, we demonstrate that excluding cyclic causal arguments leads to the concept of Aristotelian knowledge, as provided by the stable models semantics of abductive logic programs (27; 22). In Section 3.2, building upon abductive logic programming, we introduce weighted abductive logic programs as a formalism tailored to causal reasoning under uncertainty. Counterfactual reasoning within weighted abductive logic programming is further discussed in Section 3.3. We observe that weighted abductive logic programming encompasses widely-used frameworks in statistical relational artificial intelligence: Markov logic networks (46), LP$^{\mathrm{MLN}}$ programs (35), ProbLog programs (21; 26), and logic programs with annotated disjunctions (62). By embedding these formalisms into our general framework and transferring our causal reasoning there, we ensure consistent causal reasoning across the aforementioned formalisms, as elaborated in Section 3.4.

## 1.5 ProbLog programs and Counterfactual Reasoning

Lastly, we adapt the viewpoint that weighted abductive logic programming is the most general formalism for capturing (probabilistic) causal reasoning and (probabilistic) Aristotelian knowledge. If we further assert Reichenbach's common cause assumption (45) that every dependence among the modeled random variables is explained by causal reasoning, Section 3.4.3 argues that the Aristotelian knowledge enabling well-defined counterfactual reasoning corresponds to a fragment of ProbLog. In Chapter 4, we then prove, under rather mild assumptions, that the counterfactual probabilities provided by acyclic ProbLog programs are expressive enough to determine these programs up to syntactic equality. Conjecturing that this behavior persists in the cyclic case, we argue that the presented causal interpretation of ProbLog yields *the theory of probabilistic Boolean counterfactual reasoning.*

# Chapter 2

# Preliminaries

This chapter introduces the concepts that form the foundation of the theory presented in this thesis. We discuss the notions of knowledge and counterfactual. Specifically, we provide an overview of Pearl's and Bochman's theories of causality that serve as the cornerstones for our own theory. Furthermore, we briefly introduce Markov logic networks and probabilistic logic programming. These formalisms belong to the broader field of statistical relational artificial intelligence, which combines first-order logic with probabilistic reasoning under uncertainty. In Chapter 3, we then specifically use probabilistic logic programming to apply our theory of causality and counterfactual reasoning in contemporary artificial intelligence.

## 2.1 Knowledge and Counterfactuals

This thesis addresses the issue of *counterfactual reasoning*, specifically, the task of answering queries of the form:

$$\text{“Would } Q \text{ have occurred if } C \text{ had taken place before observing evidence } E\text{?”} \tag{2.1}$$

**Example 2.1.1.** Assume that we observe a fire breaking out during a thunderstorm in a house without a lightning rod. In this case, we can pose the counterfactual question: “Would the fire have occurred if we had installed a lightning rod before the thunderstorm?”

According to Lewis (37), a counterfactual statement is valid if $Q$ holds in all alternative worlds minimally deviating from the real world, where we observe the evidence $E$ in such a way that the constraint $C$ becomes true.

**Example 2.1.2.** Reconsider the scenario of Example 2.1.1. In the real world, we observe a fire that breaks out during a thunderstorm in a house without a lightning rod. According to Lewis (37), the alternative worlds should minimally deviate from this real world in such a way that the constraint of our house having a lightning rod is satisfied. Finally, the statement “The fire would not have occurred in our house had we installed a lightning rod” is equivalent to the fire not occurring in each of these alternative worlds.

However, Lewis' approach (37) leaves us with the task of constructing these alternative worlds. In this thesis, we adapt Pearl's idea (41) for the construction of these alternative worlds, which itself relies on the availability of explanations for our observations. In particular, the alternative worlds are obtained by manipulating the explanations for our observations. Thus, Pearl's counterfactual reasoning requires *knowledge* (13) or *scientific knowledge* as given by Aristotle in his *Posterior Analytics* (4). This notion of *knowledge* further builds on the theory of *deduction*, i.e., proof, presented in Aristotle's *Prior Analytics* (3). He then distinguishes between deductions and demonstrations; these are true deductions that follow the “natural order of explanation” (13).

> “By demonstration I mean a scientific deduction; and by scientific I mean one in virtue of which, by having it, we understand something. If, then, understanding is as we posited, it is necessary for demonstrative understanding in particular to depend on things which are true and primitive and immediate and more familiar than and prior to and explanatory of the conclusion (for in this way the principles will also be appropriate to what is being proved). For there will be deduction even without these conditions, but there will not be demonstration; for it will not produce understanding.” (4, APo, 71b19-25)

According to Aristotle, logically correct deductions or proofs merely result in what we call **factual knowledge**—knowledge about what is. Only demonstrations generate understanding and contribute to what Aristotle calls *knowledge* (13), or *scientific knowledge*, which we therefore refer to as **Aristotelian knowledge**. Following Sandstad (54), Angioni (1), and Bochman (10, §1.1), we adopt the viewpoint that the “natural order of explanation” (13) refers to the concept of causality. Hence, Aristotelian knowledge about a fact $A$ consists of factual knowledge about $A$, that is, that $A$ indeed takes place and a causal explanation, providing a reason *why* $A$ takes place.

**Example 2.1.3.** In general, assume that lightning strikes or arson are the only possible causes of fire. In this scenario, from the observation of a fire, we may deduce that arson occurred or a lightning strike struck our house. This deduction is true, but not a demonstration, as concluding from a fire to arson or lightning strikes goes against the order of cause and effect. In this case, we acquire factual knowledge about arson or a lightning strike, but we cannot explain why arson occurred or why a lightning strike hit our house. However, if we observe a lightning strike hitting our house and deduce that a fire is breaking out, we respect the causal order of the events. This is a demonstration, and we gain Aristotelian knowledge about the fire, understanding both that a fire is breaking out and the reason behind it, which is the lightning strike.

Our approach to counterfactual reasoning presented in Chapter 3 relies on Bochman’s formalization of Aristotelian knowledge (10, §4). Representing factual knowledge and deductions using propositional logic, Bochman formalizes causal reasoning as a binary relation on propositional formulas, which is subject to certain axioms. In particular, he assumes that causal reasoning satisfies Aquinas’ *natural necessity* (32, Ch. 1) and Leibniz’s principle of *sufficient causation* (36, p. 268), as stated below.

**Law 1** (Aquinas (2, SGA II: 35.4))**.** *“..., given the existence of the cause, the effect must necessarily follow.”*

**Law 2** (Leibniz (36, p. 268))**.** *“...there is nothing without a reason, or no effect without a cause.”*

**Remark 2.1.1.** Bochman’s representation makes no statement about whether causality is an ontological principle governing the world or an a priori category necessary to experience the world, as described by Kant (33).

The Aristotelian knowledge corresponding to our causal reasoning then consists of all states of affairs or worlds explainable by our causal reasoning. Furthermore, Bochman (10, §5) follows the idea of Descartes and specifies causal reasoning through *causal rules* (32, Ch. 1). This approach aligns Bochman’s theory of Aristotelian knowledge with Pearl’s causal models (41), enabling him to revise causal reasoning and Aristotelian knowledge based on *external interventions.*

**Example 2.1.4.** In Example 2.1.3, the installation of a lightning rod would be considered an external intervention. In particular, we require Aristotelian knowledge that certain fires are explained by lightning strikes to conclude that such a fire will not break out after the intervention of installing a lightning rod.

However, deviating from Aristotle, Bochman (10, §4) allows for cyclic and infinite arguments (39). In Section 3.1, we avoid cyclic or infinite arguments by extending Bochman’s causal theories with prior knowledge that every demonstration must start with. In Theorem 3.1.7, we then show that the resulting formalism is essentially equivalent to abductive logic programming (22) under the stable model semantics (27). As a consequence, we argue that abductive logic programming emerges as a correct framework to reason on Aristotelian knowledge. From the perspective of Pearl's causal models (41), this means that by translating a Boolean causal model to an abductive logic program, the semantics of this program precisely specifies the solutions that constitute Aristotelian knowledge.

The counterfactual reasoning introduced in this thesis further relies on expressing uncertainty about the prior knowledge that demonstrations need to start with in a probabilistic manner. In Section 3.2, therefore, we introduce our notion of probabilistic Aristotelian knowledge, represented by weighted abductive causal theories and weighted abductive logic programs. Furthermore, we follow the argument of Pearl (41) that an intervention constitutes a minimal change such that a certain constraint becomes true. Hence, in the case of (2.1), the construction of Lewis' alternative worlds should start with the same prior knowledge as the real world, where we observe the evidence $E$, and then follow the causal reasoning that results from intervening according to the constraint $C$. In particular, we first use our evidence $E$ to learn about the state of the prior knowledge, then we intervene according to the constraint $C$, and finally, we determine the probability that the query $Q$ would hold if $C$ had taken place by computing the probability of $Q$ under this modified probabilistic Aristotelian knowledge.

**Example 2.1.5.** Reconsider the Examples 2.1.1 and 2.1.3. To answer the corresponding counterfactual query, we first revise our belief in arson or a lightning strike, respectively, based on the observation of the thunderstorm. In particular, the thunderstorm makes it more likely that lightning has hit our house, causing the fire. Hence, following Example 2.1.4, we tend to believe the statement: "The fire would most likely not have occurred had we installed a lightning rod."

Finally, in Chapter 4, we address the question of whether the ability of counterfactual reasoning is equivalent to possessing probabilistic Aristotelian knowledge in our sense. First, Example 4.1.4, credited to Lifschitz by Bochman (10, Example 8.3), demonstrates that, in general, this is not the case. However, we are able to provide a reasonably large fragment of our formalism in which probabilistic Aristotelian knowledge can be uniquely reconstructed from the answers to the corresponding counterfactual queries. We conclude that from a probabilistic perspective, the situation in Example 4.1.4 is merely a borderline case and that counterfactual reasoning usually requires full Aristotelian knowledge.

## 2.2 Statistical Relational Artificial Intelligence

The field of statistical relational artificial intelligence encompasses frameworks that combine logic-based reasoning with uncertainties represented by probabilities. To introduce these frameworks, we begin by gathering the necessary prerequisites from probability theory.

### 2.2.1 Probability Theory

In this thesis, we restrict ourselves to finite probability spaces.

**Definition 2.2.1** (Probability Space)**.** A **probability space** is a tuple $(\Omega, \pi)$ consisting of a finite set $\Omega$, called **sample space**, and a **probability mass function** $\pi : \Omega \to [0, 1]$ such that

$$\sum_{\omega \in \Omega} \pi(\omega) = 1.$$

In this context, an **event** is a subset of the sample space $A \subseteq \Omega$. To every event $A \subseteq \Omega$ we assign the probability

$$\pi(A) := \sum_{\omega \in A} \pi(\omega).$$

Finally, we write

$$\pi(A_1, ..., A_n) := \pi(A_1 \cap ... \cap A_n)$$

for events $A_1, ..., A_n$, $n \in \mathbb{N}_{\geq 1}$.

**Example 2.2.1.** Let us throw a fair die. In this case, we have six possible outcomes, which we model with the sample space $\Omega := \{1, 2, 3, 4, 5, 6\}$, where $\omega \in \Omega$ means that the die shows $\omega$. As the die is supposed to be fair, we have the probability mass function $\pi : \Omega \to [0, 1]$, $\omega \mapsto 1/6$. In the probability space $(\Omega, \pi)$, we find the event $A := \{2, 4, 6\} \subseteq \Omega$ of obtaining an even number with probability $\pi(A) := 1/2$.

We primarily employ random variables to reason about events.

**Definition 2.2.2.** A **random variable** is a function $p : \Omega \to R$ and a **distribution** of $p$ is a probability space $(\Omega, \pi)$. For a **possible value** $r \in R$ we then define the event of $p$ taking the value $r$ as $\{p = r\} := p^{-1}(r)$ and denote its probability by $\pi(p = r) := \pi(p^{-1}(r))$. We call $p$ a **Boolean** random variable if $R = \{True, False\}$ and write $\pi(p) := \pi(p = True)$ as well as $\pi(\neg p) := \pi(p = False)$.

Finally, a **joint distribution** of the random variables $p_i : \Omega_i \to R_i$, $1 \leq i \leq n$ is a distribution of the random variable $\prod_i p_i : \prod_i \Omega_i \to \prod_i R_i$, $(\omega_1, ..., \omega_n) \mapsto (p_1(\omega_1), ..., p_n(\omega_n))$. Note that by projection, every joint distribution yields a distribution of the random variables $p_i$.

**Example 2.2.2.** Recall the situation in Example 2.2.1 and assume that we win one euro if the die shows an even number; otherwise, we lose one euro. This bet can be modeled with the random variable

$$p : \Omega \to \{1, -1\} \qquad \omega \mapsto \begin{cases} 1, & \text{if } \omega \in A \\ -1, & \text{otherwise} \end{cases}.$$

Here, $p(\omega) = 1$ means that we win one euro, and $p(\omega) = -1$ means we lose one euro. Specifically, for the event $A$ of Example 2.2.1, we have $A = \{p = 1\}$.

**Example 2.2.3.** Consider a field with a sprinkler in it. The grass is wet if the sprinkler is on or it rains. We model this scenario with the following Boolean random variables

$$\begin{aligned} &sprinkler : \{True, False\} \to \{True, False\}, && x \mapsto x \\ &rain : \{True, False\} \to \{True, False\}, && x \mapsto x \\ &wet : \{True, False\} \to \{True, False\}, && x \mapsto x. \end{aligned}$$

Here, $sprinkler = True$ indicates that the sprinkler is on, $rain = True$ indicates that it rains, and $wet = True$ indicates that the grass is wet. A joint distribution of these three random variables is a distribution of the random variable

$$sprinkler \times rain \times wet : \{True, False\}^3 \to \{True, False\}^3, \qquad (x, y, z) \mapsto (x, y, z).$$

For instance, we obtain a joint distribution by setting

$$\begin{aligned} \pi(False, False, False) &:= \pi(\neg sprinkler, \neg rain, \neg wet) := 0.2 \\ \pi(True, False, False) &:= \pi(sprinkler, \neg rain, \neg wet) := 0 \\ \pi(False, True, False) &:= \pi(\neg sprinkler, rain, \neg wet) := 0 \\ \pi(False, False, True) &:= \pi(\neg sprinkler, \neg rain, wet) := 0 \\ \pi(True, True, False) &:= \pi(sprinkler, rain, \neg wet) := 0 \\ \pi(True, False, True) &:= \pi(sprinkler, \neg rain, wet) := 0.3 \\ \pi(False, True, True) &:= \pi(\neg sprinkler, rain, wet) := 0.2 \\ \pi(True, True, True) &:= \pi(sprinkler, rain, wet) := 0.3. \end{aligned}$$

We find that the sprinkler is on if the first component $x$ of a tuple $(x, y, z) \in \{True, False\}^3$ is true. If we denote by $\pi_1 : \{True, False\}^3 \to \{True, False\}$, $(x, y, z) \mapsto x$, the projection to the first component, we therefore find $sprinkler = \pi_1 \circ (sprinkler \times rain \times wet)$, which shows that the sprinkler is on with a probability of

$$\begin{aligned}\pi(sprinkler) &= \pi(\pi_1 \circ (sprinkler \times rain \times wet) = True) = \\ &= \pi(sprinkler, \neg rain, \neg wet) + \pi(sprinkler, rain, \neg wet) + \\ &+ \pi(sprinkler, \neg rain, wet) + \pi(sprinkler, rain, wet) = 0.6.\end{aligned}$$

Informally, we adopt the viewpoint of Bayesianism, where the probability $\pi(A) \in [0, 1]$ of an event $A \subseteq \Omega$ represents a rational agent's degree of belief in $A$ being true. Here, "rational" means that, for any amount of money $S \in \mathbb{R}$, the agent is willing to place at most $\pi(A) \cdot S$ on the truth of $A$ in a bet with a return of $S$ (63, pp.500-501). When observing an event $B$, a rational agent is supposed to revise his beliefs as follows.

**Definition 2.2.3** (Conditional Probability)**.** Let $(\Omega, \pi)$ be a probability space and let $A, B \subseteq \Omega$ be two events. In this case, we define the probability of $A$ **conditioned** on $B$ by

$$\pi(A|B) := \begin{cases} \dfrac{\pi(A, B)}{\pi(B)} & \text{if } \pi(B) \neq 0 \\ 0, & \text{otherwise} \end{cases}.$$

**Example 2.2.4.** Assume we observe that someone wins one euro in the game described in Example 2.2.2. This observation leads to a probability of $\pi(2|p = 1) := 1/3$ for the die to show two.

Finally, conditional probabilities enable us to assess the independence of events. In particular, an event $A$ is considered independent of an event $B$ if observing $B$ does not alter our belief in $A$. This concept is formalized in the following definition.

**Definition 2.2.4** (Independence)**.** Let $(\Omega, \pi)$ be a probability space and let $A_1, \dots, A_n \subseteq \Omega$ be events for a natural number $n \in \mathbb{N}$. We say that $A_1, \dots, A_n$ are **(mutually) independent** if for any tuple of indices $(i_1, \dots, i_k) \in \{1, \dots, n\}^k$, $1 \leq k \leq n$, with $i_j \neq i_l$ for all $1 \leq j, l \leq n$, we have

$$\pi(A_{i_1}) = \pi(A_{i_1} | A_{i_2}, \dots, A_{i_k}) \text{ or equivalently } \pi(A_{i_1}, \dots, A_{i_k}) = \prod_{i=1}^{k} \pi(A_{i_k}).$$

Otherwise, we call the events $A_1, \dots, A_n$ **(mutually) dependent**.

**Example 2.2.5.** Consider throwing a second die in the situation described in Example 2.2.1. Now, we have 36 possible outcomes $\Omega := \{(1, 1), ..., (6, 6)\}$, where $(i, j) \in \Omega$ signifies that the first die shows $i$ while the second die shows $j$. Assuming that the result of the first die does not influence the outcome of the second die, we have $\pi((i, j)) := \pi(i) \cdot \pi(j) := 1/36$. This reflects the independence of the events of the first die showing $i$ and the second die showing $j$.

### 2.2.2 Propositional Logic and LogLinear Models

Next, we introduce propositional logic, the simplest form of logic, as a formalism to reason about Boolean functions.

#### Propositional Logic

In this thesis, we employ propositional logic to reason about truth, i.e. to specify sets of Boolean functions that satisfy certain constraints. However, to reason about Boolean functions, we first need to agree on the domain of these functions.

**Definition 2.2.5** (Propositional Alphabet)**.** A **propositional alphabet** $\mathfrak{P}$ is a finite set whose elements we call **propositions**.

**Example 2.2.6.** Assume that a road passes through a field with a sprinkler in it. The sprinkler is activated by a weather sensor when it is not cloudy. In addition, the road pavement is wet when the sprinkler is on or when it rains. Lastly, when the road is wet, it becomes slippery.

Next, *cloudy* denotes the fact that the weather is cloudy, *sprinkler* denotes the fact that the sprinkler is on, *rain* denotes that it is raining, *wet* denotes that the pavement of the road is wet, and *slippery* means that the road is slippery. Hence, we gave a propositional alphabet

$$\mathfrak{P} := \{cloudy, sprinkler, rain, wet, slippery\}$$

to reason on the described scenario.

Fix a propositional alphabet $\mathfrak{P}$. As mentioned earlier, our objective is to reason on the truth of the propositions in $\mathfrak{P}$.

**Definition 2.2.6** (Structure)**.** A **($\mathfrak{P}$-)structure** $\omega : \mathfrak{P} \to \{True, False\},\ p \mapsto p^{\omega}$ is a Boolean function on $\mathfrak{P}$. It is important to note that such a function can be safely identified with the set of all propositions $p \in \mathfrak{P}$ that are true in $\omega$, i.e., with the set $\omega := \{p \in \mathfrak{P}\colon\ p^{\omega} = True\}$.

**Example 2.2.7.** In the situation of Example 2.2.6, a structure $\omega$ is a complete state description such as

$$\begin{aligned} &cloudy \mapsto True && sprinkler \mapsto False \\ &rain \mapsto True && wet \mapsto True \\ &slippery \mapsto True. \end{aligned}$$

Next, in formulas, we specify constraints on the structures under consideration.

**Definition 2.2.7** (Propositional Formulas)**.** The **(propositional) formulas** in the alphabet $\mathfrak{P}$ or **($\mathfrak{P}$)-formulas** are recursively defined by the following axioms:

i) The **falsity** symbol $\bot \notin \mathfrak{P}$ and the **truth** symbol $\top \notin \mathfrak{P}$ are formulas.

ii) Every proposition $p \in \mathfrak{P}$ is an **atomic** formula or **atom**.

iii) If $\phi$ is a formula, then $\neg\phi$ is also a formula.

iv) If $\phi$ and $\psi$ are formulas, then $(\phi \wedge \psi)$ is also a formula.

A **literal** is a formula $l \in \{p, \neg p\}$ for an **underlying proposition** $p \in \mathfrak{P}$. We refer to $l$ as a **positive** literal if it is of the form $p \in \mathfrak{P}$ and as a **negative** literal otherwise.

**Remark 2.2.1.** In situations where there is no possible confusion about the meaning of a formula $\phi$, we denote $\phi$ omitting parentheses ( and ).

**Example 2.2.8.** In the alphabet of Example 2.2.6, we can form the formulas

$$sprinkler,\ \neg cloudy \text{ and } (sprinkler \wedge \neg cloudy).$$

Here, *sprinkler* is an atom or (positive) literal, whereas $\neg cloudy$ is a (negative) literal. Finally, the formula $(sprinkler \wedge \neg cloudy)$ is neither an atom nor a literal. As no confusion occurs if we omit parentheses, we further denote the last formula by $sprinkler \wedge \neg cloudy$.

Further, we define the meaning or semantics of a propositional formula.

**Definition 2.2.8** (Models Relation)**.** Whether a structure $\omega$ **satisfies** or **models** a formula $\phi$, denoted by $\omega \models \phi$, is defined recursively based on Definition 2.2.7.

i) We find $\omega \not\models \bot$ and $\omega \models \top$, i.e., $\omega \models \bot$ is not valid for any structure $\omega$.

ii) For every atomic formula $p \in \mathfrak{P}$ we find $\omega \models p$ if and only if $p^\omega = True$, that is, $p \in \omega$.

iii) For every formula $\phi$ we find $\omega \models \neg\phi$ if and only if $\omega \not\models \phi$, i.e., if $\omega \models \phi$ is not valid.

iv) For every two formulas $\phi$ and $\psi$ we find $\omega \models (\phi \wedge \psi)$ if and only if $\omega \models \phi$ and $\omega \models \psi$.

Next, the structure $\omega$ **satisfies** or **models** a set of formulas $\Phi$, written $\omega \models \Phi$, if and only if we find $\omega \models \phi$ for every formula $\phi \in \Phi$. In this case, we also say that $\omega$ is a **model** of $\Phi$.

**Remark 2.2.2.** We also associate a structure $\omega : \mathfrak{P} \to \{True, False\}$ with the set of literals it renders true, that is, with $\{p \in \mathfrak{P}\colon\ p^\omega = True\} \cup \{\neg p\colon\ p \in \mathfrak{P},\ p^\omega = False\}$.

**Example 2.2.9.** In the case of the structure $\omega$ in Example 2.2.7, we find $\omega \not\models \bot$, $\omega \models \top$, $\omega \models \mathit{cloudy}$, $\omega \models \neg \mathit{sprinkler}$, $\omega \models (\mathit{cloudy} \wedge \neg\mathit{sprinkler})$ and $\omega \not\models (\mathit{wet} \wedge \neg\mathit{slippery})$.

**Notation 2.2.1.** For two formulas $\phi$ and $\psi$ we use the following abbreviations.

i) We write $(\phi \vee \psi)$ for $\neg(\neg\phi \wedge \neg\psi)$.

ii) We write $(\phi \to \psi)$ for $(\neg\phi \vee \psi)$.

iii) We write $(\phi \leftarrow \psi)$ for $(\psi \to \phi)$.

iv) We write $(\phi \leftrightarrow \psi)$ for $((\phi \to \psi) \wedge (\psi \to \phi))$.

**Example 2.2.10.** In the scenario of Example 2.2.6, $(\mathit{cloudy} \vee \mathit{slippery})$ means that it is cloudy or the road is slippery. Similarly, $(\mathit{rain} \to \mathit{wet})$ means that the road is wet if it rains and $(\mathit{wet} \leftrightarrow \mathit{slippery})$ means that the road is slippery if and only if it is wet.

**Remark 2.2.3.** Again, we denote a formula $\phi$ by omitting parentheses ( and ) if no confusion is possible.

Finally, we also would like to express that a formula subsumes another formula.

**Definition 2.2.9** (Semantic Entailment)**.** A formula $\phi$ **(semantically) entails** a formula $\psi$, written $\phi \models \psi$, if whenever $\omega \models \phi$ we also find $\omega \models \psi$, i.e. if every structure modeling $\phi$ also models $\psi$. Analogously, a set of formulas $\Phi$ **(semantically) entails** a formula $\psi$, written $\Phi \models \psi$, if for every structure $\omega$ we find $\omega \models \psi$ whenever $\omega \models \Phi$. We call $\Phi$ **consistent** if $\Phi$ has at least one model, i.e. if $\Phi \not\models \bot$.

A set of formulas $\Phi$ is **deductively closed** if whenever $\Phi \models \psi$ we find $\psi \in \Phi$. The **deductive closure** of a set of formulas $\Phi$ is the smallest deductively closed set $\bar{\Phi}$ such that $\Phi \subseteq \bar{\Phi}$. Finally, we call a consistent deductively closed set of formulas $\Phi$ a **world** if $\Phi$ is maximal with respect to the subset relation $(\subseteq)/2$.

**Notation 2.2.2.** We observe that every world $\Phi = \bar{\mathbf{L}}$ is the deductive closure of the set of its literals $\mathbf{L} := \{l \in \{p, \neg p\} : p \in \mathfrak{P},\ l \in \Phi\}$. As $\Phi$ is consistent and maximal regarding the subset relation $(\subseteq)/2$, we further find that $\mathbf{L} := (\mathbf{L} \cap \mathfrak{P}) \cup \{\neg p\colon\ p \in \mathfrak{P}, p \notin \Phi\}$. Hence, we can identify $\Phi$ with the set of propositions $\mathbf{L} \cap \mathfrak{P}$.

As we also identify structures with sets of propositions in Definition 2.2.6, we shall henceforth interchangeably use the terms “structure”, “subset of propositions”, and “world”. Specifically, we treat worlds and structures as partially ordered by the subset relation $(\subseteq)/2$.

**Example 2.2.11.** In the alphabet $\mathfrak{P}$ of Example 2.2.6, we find $(\mathit{cloudy} \wedge \neg\mathit{sprinkler}) \models \neg\mathit{sprinkler}$ and $(\mathit{cloudy} \wedge \neg\mathit{sprinkler}) \not\models \mathit{rain}$.

However, we wish to reason not only on formulas, but also on their certainty.

### LogLinear Models

Fix a joint distribution $\pi$ on a propositional alphabet $\mathfrak{P}$. Furthermore, assume that the relevant information about $\pi$ is contained in the $n$ probabilities $\pi(\phi_i) \in (0,1)$ of mutually exclusive formulas $\phi_i$, where $1 \leq i \leq n$ and $n \in \mathbb{N}$. Here, this means that the distribution $\pi$ results from the following principle of indifference:

**Law 3** (Principle of Indifference)**.** *We consider two possible values of a random variable p as equally probable when we have no reason to prefer one value over the other.*

**Example 2.2.12.** Let us throw a die. We denote by 1, 2, 3, 4, 5, and 6 that the die shows one, two, three, four, five, and six, respectively, resulting in the alphabet

$$\mathfrak{P} := \{1, 2, 3, 4, 5, 6\}.$$

If we have no further information about the die, according to the principle of indifference in Law 3, we consider the die to be fair, i.e., we assume that throwing the die results in $i$ with a probability of $\pi(i) := 1/6$ for all $1 \leq i \leq 6$.

Assume we are further informed that the die is biased and shows 1 or 2 with probability 1/2, that is, $\pi(1 \vee 2) = 1/2$, and 6 with probability 1/10, i.e., $\pi(6) = 1/10$. According to the principle of indifference, this results in the following probabilities:

$$\begin{array}{lll} \pi(1) = 1/4 & \pi(2) = 1/4 & \pi(3) = 4/30 \\ \pi(4) = 4/30 & \pi(5) = 4/30 & \pi(6) = 1/10 \end{array}$$

In particular, we first distribute the probability $\pi(1 \vee 2) = 1/2$ uniformly to the events 1 and 2 and set $\pi(6) := 1/10$. Finally, we distribute the remaining probability mass of $1 - \pi(1 \vee 2) - \pi(6) = 4/10$ uniformly to the remaining events 3, 4, and 5.

Unfortunately, we need the mutual exclusivity of the formulas $\phi_i$, $1 \leq i \leq n$, to apply the principle of indifference in Law 3 for extending the probabilities $\pi(\phi_i)$ to a distribution on the alphabet $\mathfrak{P}$.

**Example 2.2.13.** Recall the situation of Example 2.2.12 and suppose we learn that the die shows 1 or 2 with probability $\pi(1 \vee 2) = 1/3$. Further, assume we also find out that the die shows 2 or 3 with probability $\pi(2 \vee 3) = 1/3$. Now, the principle of indifference in Law 3 does not tell us how to merge the information about throwing 2, provided by the probabilities $\pi(1 \vee 2) = 1/3$ and $\pi(2 \vee 3) = 1/3$, to get a joint distribution $\pi$ on $\mathfrak{P}$.

Suppose that the relevant information about the joint distribution $\pi$ on the alphabet $\mathfrak{P}$ is contained in the $n$ probabilities $\pi(\phi_i) \in (0,1)$ of the formulas $\phi_i$, where $1 \leq i \leq n$ and $n \in \mathbb{N}$, which are no longer assumed to be mutually exclusive. This now means that the distribution $\pi$ results from maximizing the **entropy**

$$H(\pi) := \sum_{p \in \mathfrak{P}} (-\ln(\pi(p))) \cdot \pi(p)$$

under the constraint that the formulas $\phi_i$ hold with probability $\pi(\phi_i)$ for $1 \leq i \leq n$.

Since the function $-\ln : (0,1] \to [0,\infty)$ is strictly monotonically decreasing, with $-\ln(0) = \infty$ and $-\ln(1) = 0$, we interpret $-\ln(\pi(p))$ as our degree of surprise about the proposition $p \in \mathfrak{P}$. In particular, the number $-\ln(\pi(p))$ becomes large for small probabilities $\pi(p)$; that is, the more we are surprised about $p$ being true, the larger the number $-\ln(\pi(p))$ becomes. Hence, we interpret $H(\pi)$ as the average degree of surprise in a distribution $\pi$. We further assume that the more we are surprised by our observations, the more randomness is captured in $\pi$, and that randomness stands for missing information. In fact, entropy $H(\pi)$ is known to measure missing information in a distribution $\pi$ (57).

**Remark 2.2.4.** Note that $-\ln : (0, 1] \to [0, \infty)$ is used in the definition of entropy, as one wants the entropy to be additive regarding the distributions of independent random variables.

Given that we only know about the probabilities $\pi(\phi_1), \ldots, \pi(\phi_n)$, it makes a priori sense to assume the distribution $\pi$ on the propositions $\mathfrak{P}$ that extends the probabilities $\pi(\phi_1), \ldots, \pi(\phi_n)$ by maximizing the missing information, that is, the entropy $H(\pi)$. Maximizing entropy also resembles the principle of indifference in Law 3 if the formulas $\phi_i$ are mutually exclusive (19).

However, in general, maximizing the entropy does not yield a distribution that can be easily described using the probabilities $\pi(\phi_i)$, $1 \leq i \leq n$. Nevertheless, as the distribution $\pi$ is essentially determined by giving one number for every formula $\phi_i$, $1 \leq i \leq n$, one aims for a parameterization of $\pi$ that is also given by a number $w_i \in \mathbb{R}$ for every formula $\phi_i$, $1 \leq i \leq n$. Here, we apply the following result of Berger et al. (8).

**Parametrization 4** (Berger et al. (8))**.** *We find $n$* ***degrees of certainty*** *$w_i \in \mathbb{R}$, $1 \leq i \leq n$ such that $\pi$ is given by setting*

$$\pi(\omega) := \frac{\exp\left(\sum\limits_{\omega \models \phi_i} w_i\right)}{\sum\limits_{\omega' \text{ world}} \exp\left(\sum\limits_{\omega' \models \phi_i} w_i\right)}$$

*for every world $\omega$.*

Finally, a LogLinear model of Richardson and Domingos (46) formalizes a set of formulas with assigned degrees of certainty in the sense of Berger et al. (8).

**Definition 2.2.10** (LogLinear Models)**.** A **LogLinear model (with binary features)** is a set $\Phi$ consisting of **weighted constraints** $(w, \phi)$, where $w \in \mathbb{R} \cup \{\infty\}$ is a real weight and $\phi$ is a formula.

**Example 2.2.14.** Recall the situation in Example 2.2.6 and consider a road that passes through a field with a sprinkler in it. We denote by *cloudy* that the weather is cloudy. The sprinkler may be switched on, denoted *sprinkler*, by a weather sensor in the event of sunny weather, that is, if $\neg$*cloudy* holds. By *rain* we denote that it is raining and by *wet* we denote that the pavement of the road is wet, which happens if it rains or if the sprinkler is on. Finally, the road is slippery, denoted *slippery*, if it is wet. Hence, we consider the propositional alphabet

$$\mathfrak{P} := \{cloudy, sprinkler, rain, wet, slippery\}$$

to reason on the described scenario, which might lead to the LogLinear model

$$\Phi := \{(\ln(2), cloudy \to rain), (\ln(3), \neg cloudy \to sprinkler), (\infty, wet \leftrightarrow rain)\}.$$

Parametrization 4 then yields the following semantics for LogLinear models.

**Definition 2.2.11** (Semantics of LogLinear Models)**.** Given a LogLinear model $\Phi$, a **possible world** $\omega$ is a world that models each **hard constraint** $(\infty, \phi) \in \Phi$, i.e., $\omega \models \phi$ whenever $(\infty, \phi) \in \Phi$. We associate with every possible world $\omega$ the **weight**

$$w_\Phi(\omega) := w(\omega) := \prod_{\substack{(w,\phi)\in\Phi \\ w\neq\infty \\ \omega\models\phi}} \exp(w) = \exp\left(\sum_{\substack{(w,\phi)\in\Phi \\ w\neq\infty \\ \omega\models\phi}} w\right)$$

and set $w(\omega) = 0$ for every world $\omega$ that is not a possible world. Further, we define the **weight** of a formula $\phi$ to be

$$w(\phi) := \sum_{\substack{\omega \text{ world} \\ \omega \models \phi}} w(\omega).$$

Finally, we interpret weights as degrees of certainty and assign to each world or formula _ the **probability**

$$\pi_\Phi(\_) := \pi(\_) := \frac{w(\_)}{w(\top)}.$$

**Remark 2.2.5.** Let $\Phi$ be a LogLinear model. Upon committing to Parametrization 4, there exists no intuitive interpretation of the weighted constraints $(w, \phi) \in \Phi$ with real weights $w \in \mathbb{R}$. Only hard constraints $(\infty, \phi) \in \Phi$ encode that a formula $\phi$ necessarily holds.

**Example 2.2.15.** In the situation of Example 2.2.14, we find $\pi(\mathit{rain}|\mathit{cloudy}) := 2/3$ as well as $\pi(\mathit{sprinkler}|\neg \mathit{cloudy}) := 3/4$. Furthermore, we deduce that the road is slippery if and only if it is wet.

### 2.2.3 Relational First-Order Logic and Markov Logic Networks

When modeling with propositional alphabets, propositions often indicate that elements in a domain satisfy specific properties. Next, we aim to extend our language to express such properties.

**Example 2.2.16.** Consider a group $G := \{p_1, ..., p_n\}$, which regularly meets in a pub. Each member of $G$ may start smoking spontaneously, denoted $\mathit{spontaneously\_smokes}(p_i)$, where $1 \leq i \leq n$. Furthermore, a group member $p_i \in G$ smokes, denoted $\mathit{smokes}(p_i)$, if he either smokes spontaneously or if there is another smoking group member $p_j \in G$, i.e., $\mathit{smokes}(p_j)$. This behavior is expressed by the following propositional formulas

$$\mathit{spontanously\_smokes}(p_i) \to \mathit{smokes}(p_i), \; 1 \leq i \leq n \tag{2.2}$$

$$\mathit{smokes}(p_i) \to \mathit{smokes}(p_j), \; 1 \leq i, j \leq n \tag{2.3}$$

Since all formulas of the form (2.2) and (2.3) only differ in the mentioned group members, we would like to summarize them with the templates

$$\mathit{spontanously\_smokes}(X) \to \mathit{smokes}(X) \tag{2.4}$$

$$\mathit{smokes}(X) \to \mathit{smokes}(Y) \tag{2.5}$$

for variables $X$ and $Y$.

We identify templates like (2.4) and (2.5) with universally closed formulas in prenex normal form that can be formulated within relational alphabets. This fragment of first-order logic is characterized by allowing no function symbols, no existential quantifiers, only permitting finitely many constants and predicate symbols, and restricting universal quantifiers to the beginning of formulas. With this limitation, we can employ the grounding method to translate our fragment of first-order logic into propositional logic, facilitating the computation of Herbrand models for a given formula.

In this way, we denote constraints that are symmetric with respect to permutations of domain elements. Introducing these symmetries in LogLinear models not only speeds up computations, but also facilitates the learning task. Specifically, assuming that domain elements behave in the same way shrinks the search space of possible models for a given set of data.

### Relational First-Order Logic

Formally, we express the properties of elements in a domain, such as smoking in Example 2.2.16, with relational alphabets.

**Definition 2.2.12** (Relational Alphabet (24, §1B, p. 4))**.** A **variable** is an expression of the form $X_j$ for a natural number $j \in \mathbb{N}$. Further, a **(relational) alphabet** is a tuple $\mathfrak{A} := (\mathcal{C}, \mathcal{P})$ consisting of

a finite set of **constants** $\mathcal{C} := \mathcal{C}(\mathfrak{A}) = \{c_1, ..., c_l\}$ and

a finite set of **predicates** $\mathcal{P} := \mathcal{P}(\mathfrak{A}) = \{p_1, ..., p_m\}$, each equipped with an **arity** $\text{ar}(p_i) \in \mathbb{N}_{\geq 0}$.

Furthermore, the sets $\mathcal{C}$ and $\mathcal{P}$ are disjoint with $X_j \notin \mathcal{C}$ and $X_j \notin \mathcal{P}$ for all $j \in \mathbb{N}$.

**Remark 2.2.6.** We do not allow function symbols in relational alphabets.

**Example 2.2.17.** To represent the scenario in Example 2.2.16, we introduce the constants

$$\mathcal{C} := G = \{p_1, ..., p_n\}.$$

Further, we introduce predicates $\mathcal{P} := \{\mathit{spontaneously\_smokes}, \mathit{smokes}\}$, where both predicates *spontaneously_smokes* and *smokes* have arity one.

**Notation 2.2.3.** In this thesis, we refer to variables $X_j$, $j \in \mathbb{N}$ by strings starting with an uppercase letter.

Fix a relational alphabet $\mathfrak{A} := (\mathcal{C}, \mathcal{P})$. We use the alphabet $\mathfrak{A}$ to reason on relations in the domain of constants $\mathcal{C}$.

**Definition 2.2.13** (Atoms and Ground Atoms)**.** An **atom** is an expression $A := p(t_1, ..., t_n)$, where $p \in \mathcal{P}$ is a predicate of arity $\text{ar}(p) := n$ and where $t_i$ is a variable or constant for $1 \leq i \leq n$. Furthermore, we say that $A$ is **ground** if $t_1, ..., t_n \in \mathcal{C}$ are constants. A **(Herbrand) structure** $\omega$ is then a set of ground atoms, representing the set of all valid relations, i.e., we consider $p(t_1, ..., t_n)$ as valid if $p(t_1, ..., t_n) \in \omega$. Note that a Herbrand structure $\omega$ also denotes a structure of the propositional alphabet $\mathfrak{P}(\mathfrak{A})$ that consists of all ground atoms in the alphabet $\mathfrak{A}$.

**Remark 2.2.7.** As there are only finitely many predicates and constants in the alphabet $\mathfrak{A}$, it gives rise to only finitely many ground atoms. Therefore, the propositional alphabet $\mathfrak{P}(\mathfrak{A})$ is a finite set. In particular, we observe that every Herbrand structure $\omega \subseteq \mathfrak{P}(\mathfrak{A})$ is a finite set.

**Example 2.2.18.** In Example 2.2.17, we obtain the atoms $\mathit{smokes}(p_1)$ and $\mathit{smokes}(X_1)$. The expression $\mathit{smokes}(p_1)$ is a ground atom, whereas the expression $\mathit{smokes}(X_1)$ is not.

Further, we can form the following Herbrand structures:

$$\begin{aligned}
&\omega_1 := \emptyset \\
&\omega_2 := \{\mathit{spontaneously\_smokes}(p_1), \mathit{smokes}(p_1), ..., \mathit{smokes}(p_n)\} \\
&\omega_3 := \{\mathit{smokes}(p_1)\}
\end{aligned}$$

Here, $\omega_1$ denotes the situation where nobody in the group $G$ smokes. The structure $\omega_2$ means that $p_1$ starts to smoke spontaneously and all members of the group $p_1, ..., p_n$ smoke. Lastly, $\omega_3$ means that no one starts smoking spontaneously and we find that the group member $p_1$ smokes.

First-order formulas are now used to reason on Herbrand structures.

**Definition 2.2.14** (First-Order Formula)**.** A **first-order formula** $\phi$ with **free variables** $\text{var}(\phi)$ is recursively defined by the following axioms.

i) The **falsity** symbol $\bot$ and the **truth** symbol $\top$ are first-order formulas with

$$\text{var}(\bot) := \text{var}(\top) := \emptyset.$$

ii) Every atom $p(t_1, ..., t_n)$ is a first-order formula with

$$\mathrm{var}(p(t_1, ..., t_n)) := \{t_i : t_i \text{ variable}, 1 \leq i \leq n\}.$$

iii) If $\phi$ is a first-order formula, then $\neg\phi$ is also a first-order formula. In this case, we set

$$\mathrm{var}(\neg\phi) := \mathrm{var}(\phi).$$

iv) If $\phi$ and $\psi$ are first-order formulas, also $(\phi \wedge \psi)$ is. In this case, we set

$$\mathrm{var}((\phi \wedge \psi)) := \mathrm{var}(\phi) \cup \mathrm{var}(\psi).$$

v) If $\phi$ is a first-order formulas with a free variable $X \in \mathrm{var}(\phi)$, also $(\forall_X \phi)$ is a first-order formula with free variables

$$\mathrm{var}((\forall_X \phi)) := \mathrm{var}(\phi) \setminus \{X\}.$$

**Notation 2.2.4.** Once again, we employ the abbreviations in Notation 2.2.1 to represent first-order formulas in the relational alphabet $\mathfrak{A}$. In addition, one usually writes $(\exists_X \phi)$ for a formula of the form $\neg(\forall_X \phi)$. Furthermore, if there is no possible confusion about the meaning of a formula $\phi$, we denote $\phi$ omitting parentheses ( and ).

**Example 2.2.19.** In the alphabet of Example 2.2.17, we can construct the following first-order formulas.

$$\begin{aligned}
&\phi_1 := \forall_X \neg smokes(X) \\
&\phi_2 := \forall_X spontaneously\_smokes(X) \to smokes(X) \\
&\phi_3 := \forall_X smokes(X) \to smokes(Y)
\end{aligned}$$

The formula $\phi_1$ then means that nobody smokes, $\phi_2$ means that everybody who starts smoking spontaneously actually smokes. Finally, $\phi_3$ has no intended meaning. However, when replacing $Y$ with a constant $p_i$ for $1 \leq i \leq n$, $\phi_3$ leads to the restriction that the group member $p_i$ smokes if there is another member of the group that smokes.

We employ ground substitutions to assign a truth value to an arbitrary first-order formula $\phi$.

**Definition 2.2.15** (Ground Substitution and Herbrand Model)**.** A **ground substitution** is a function $\gamma : \{X_j\}_{j \in \mathbb{N}} \to \mathcal{C}, V \mapsto V^\gamma$ that assigns a constant to every variable. For a variable $X$ and a constant $c \in \mathcal{C}$ we further define the ground substitution

$$\gamma[c/X] : \{X_j\}_{j \in \mathbb{N}} \to \mathcal{C}, \qquad X_j \mapsto \begin{cases} X_j^\gamma, & X_j \neq X \\ c, & X_j = X \end{cases}.$$

Finally, we extend every ground substitution $\gamma$ to the set of constants $\mathcal{C}$ by setting $c^\gamma := c$ for every constant $c \in \mathcal{C}$. Whether a Herbrand structure $\omega$ **satisfies** a first-order formula $\phi$ under a ground substitution $\gamma$, written $(\omega, \gamma) \models \phi$, is then recursively defined as follows:

i) We find $(\omega, \gamma) \not\models \bot$ and $(\omega, \gamma) \models \top$.

ii) For every atom $p(t_1, ..., t_n)$ we find $(\omega, \gamma) \models p(t_1, ..., t_n)$ if and only if $p\,(t_1^\gamma, ..., t_n^\gamma) \in \omega$.

iii) For every formula $\phi$ we find $(\omega, \gamma) \models \neg\phi$ if and only if $(\omega, \gamma) \not\models \phi$.

iv) For every two formulas $\phi$ and $\psi$ we find $(\omega, \gamma) \models (\phi \wedge \psi)$ if and only if $(\omega, \gamma) \models \phi$ and $(\omega, \gamma) \models \psi$.

v) For every formula $\phi$ with a free variable $X \in \mathrm{var}(\phi)$ we find $(\omega, \gamma) \models (\forall_X \phi)$ if and only if $(\omega, \gamma[c/X]) \models \phi$ for every constant $c \in \mathcal{C}$.

Let $\phi$ be a formula that does not mention any free variables, that is, $\text{var}(\phi) = \emptyset$. In this case, for any Herbrand structure $\omega$, we observe either $(\omega, \gamma) \models \phi$ or $(\omega, \gamma) \not\models \phi$ independently of the ground substitution $\gamma$. In the former case, we say that $\omega$ is a **Herbrand model** of $\phi$ and write $\omega \models \phi$.

**Remark 2.2.8.** Going back to Tarski (58), one usually associates first-order formulas with more general models, also capturing relations on arbitrary domains and not only on the domains of constants (12, Definition 32).

**Example 2.2.20.** Recall the formula

$$\phi_3 := \forall_X \mathit{smokes}(X) \to \mathit{smokes}(Y)$$

from Example 2.2.19, and let $\gamma : \{X_j\}_{j\in\mathbb{N}} \to \{p_1, ..., p_n\}$ be a ground substitution with $Y^\gamma = p_j$. A Herbrand structure $\omega$ now satisfies $\phi_3$ under the ground substitution $\gamma$ if and only if

$$(\omega, \gamma[p_i/X]) \models \mathit{smokes}(p_i) \to \mathit{smokes}(p_j)$$

for all $1 \leq i \leq n$, that is, we find $p_j$ smoking in $\omega$ if there exists another smoking group member.

However, in this thesis, we do not study arbitrary first-order formulas. Instead, we restrict our interest to the following fragment of relational first-order logic that can be translated into propositional logic.

**Definition 2.2.16** (Quantifier-Free Formula, Universal Formula, Universally Closed Formula)**.** A **quantifier-free formula** $\phi$ with **(free) variables** $\text{var}(\phi)$ is recursively defined by Axioms i) – iv) of Definition 2.2.14:

i) The falsity symbol $\bot$ and the truth symbol $\top$ are quantifier-free formulas with

$$\text{var}(\bot) := \text{var}(\top) := \emptyset.$$

ii) Every atom $p(t_1, ..., t_n)$ is a quantifier-free formula with

$$\text{var}(p(t_1, ..., t_n)) := \{t_i : t_i \text{ variable}, 1 \leq i \leq n\}.$$

iii) If $\phi$ is a quantifier-free formula, then $\neg\phi$ is also a quantifier-free formula with

$$\text{var}(\neg\phi) := \text{var}(\phi).$$

iv) If $\phi$ and $\psi$ are quantifier-free formulas, also $(\phi \wedge \psi)$ is. In this case, we set

$$\text{var}((\phi \wedge \psi)) := \text{var}(\phi) \cup \text{var}(\psi).$$

A **literal** $L$ is a quantifier-free formula $A$ or $\neg A$ for an atom $A$. We call $L$ a **positive** literal if it is of the form $A$ and a **negative** literal otherwise.

A **universal formula** (in prenex normal form) is a first-order formula $\psi$ of the form

$$\forall_{Y_1} ... \forall_{Y_n} \phi$$

for a quantifier-free formula $\phi$. Further, we call $\psi$ a **universally closed** formula if additionally $\text{var}(\psi) = \emptyset$, i.e., $\text{var}(\phi) = \{Y_1, ..., Y_n\}$. In this case, we also call $\psi$ the **universal closure** of the quantifier-free formula $\phi$. As we find $\text{var}(\psi) = \emptyset$ for every universally closed formula $\psi$, we find that a Herbrand structure $\omega$ is a **Herbrand model** of $\psi$ if $\omega \models \psi$ in Definition 2.2.17.

**Remark 2.2.9.** Universal formulas are usually defined in a more elaborate way (12, Definition 14). However, every such formula can be translated into a universal formula in prenex normal form as defined in Definition 2.2.16 with the same models in the sense of Remark 2.2.8 (12, Theorem 37).

For universally closed formulas, a model as outlined in Remark 2.2.8 exists if and only if there exists a Herbrand model as defined in Definition 2.2.17 (12, Corollary 65). Hence, if we want to check whether a given universally closed formula is satisfiable, i.e. whether it possesses a model in the sense of Remark 2.2.8, it is sufficient to check whether it has a Herbrand model.

As we only allow for finitely many predicates and constants, there are only finitely many Herbrand structures, making a search for a Herbrand model accessible for computation. This is a classical motivation for considering Herbrand models in computer science.

**Example 2.2.21.** In the alphabet of Example 2.2.17, we can construct the following quantifier-free formulas.

$$\begin{aligned}
&\phi_1 := \neg \mathit{smokes}(X)\\
&\phi_2 := \mathit{spontaneously\_smokes}(X) \to \mathit{smokes}(X)\\
&\phi_3 := \mathit{smokes}(X) \to \mathit{smokes}(Y)
\end{aligned}$$

The formula $\phi_1$ is a negative literal, while $\phi_2$ and $\phi_3$ are not literals. They give rise to the following universally closures:

$$\begin{aligned}
&\psi_1 := \forall_X \neg \mathit{smokes}(X)\\
&\psi_2 := \forall_X (\mathit{spontaneously\_smokes}(X) \to \mathit{smokes}(X))\\
&\psi_3 := \forall_X \forall_Y (\mathit{smokes}(X) \to \mathit{smokes}(Y))
\end{aligned}$$

Here, $\psi_1$ means that nobody is smoking, $\psi_2$ means that everybody who starts smoking spontaneously actually smokes, and $\psi_3$ means that somebody smokes if there is another smoking group member.

Quantifier-free formulas, respectively, universally closed formulas, now yield the desired templates. Using the grounding method, they can be translated into propositional logic, facilitating the computation of Herbrand models.

**Definition 2.2.17** (Ground Instances)**.** Let $\phi$ be a quantifier-free formula. To every ground substitution $\gamma$ we associate the **ground instance** $\phi^\gamma$ by recursion over Definition 2.2.16:

i) For the falsity symbol $\bot$ and the truth symbol $\top$ we have $\bot^\gamma := \bot$ and $\top^\gamma := \top$.

ii) For an atomic formula $p(t_1, ..., t_n)$ we have $p(t_1, ..., t_n)^\gamma := p\left(t_1^\gamma, ..., t_n^\gamma\right)$.

iii) If $\phi$ is a formula, we have $(\neg\phi)^\gamma := \neg\phi^\gamma$.

iv) If $\phi$ and $\psi$ are formulas, we find $(\phi \wedge \psi)^\gamma := (\phi^\gamma \wedge \psi^\gamma)$.

Finally, let $\phi$ be a quantifier-free formula. The **grounding** $\bar{\phi}$ of the formula $\phi$ is the set of all ground instances $\phi^\gamma$ of $\phi$. If $\Phi$ is a set of quantifier-free formulas, we further define the **grounding** of $\Phi$ as the set $\bar{\Phi} := \bigcup_{\phi\in\Phi} \bar{\phi}$.

**Example 2.2.22.** In Example 2.2.21, we find that the formula $\phi_3 := \mathit{smokes}(X) \to \mathit{smokes}(Y)$ has the ground instances

$$\mathit{smokes}(p_i) \to \mathit{smokes}(p_j),\ 1 \le i, j \le n. \tag{2.6}$$

Let $\psi = \forall_{Y_1} ... \forall_{Y_n} \phi$ be the universal closure of the quantifier-free formula $\phi$. Grounding now enables us to translate the formula $\psi$ to a finite set of propositional formulas:

As the alphabet $\mathfrak{A}$ mentions only finitely many predicates and constants, the grounding $\bar{\phi}$ gives rise to a finite set of propositional formulas in the alphabet $\mathfrak{P}(\mathfrak{A})$ consisting of all ground atoms in $\mathfrak{A}$. One then verifies that for every ground substitution $\gamma$ and for every Herbrand structure $\omega$ we find $(\omega, \gamma) \models \phi$ if and only if $\omega \models \phi^\gamma$ in propositional logic. In particular, we find $\omega \models \psi$ if and only if $\omega \models \bar{\phi}$ in propositional logic. Translating universally closed formulas in prenex normal form to a finite set of formulas in a finite propositional alphabet facilitates the computation of Herbrand models.

**Example 2.2.23.** In Example 2.2.21, we find that the Herbrand models of the universally closed formula

$$\psi_3 := \forall_X \forall_Y \mathit{smokes}(X) \to \mathit{smokes}(Y)$$

are exactly given by the $\mathfrak{P}(\mathfrak{A})$-structures $\omega$ that satisfy all the formulas (2.6) in propositional logic, that is, in the sense of Definition 2.2.8. Again, we denote by $\mathfrak{P}(\mathfrak{A})$ the set of all ground atoms.

**Remark 2.2.10.** Given a finite set of universally closed formulas in prenex normal form, by construction, these formulas mention only finitely many variables. Hence, once the formulas are fixed, we may also commit ourselves to finitely many variables $X_1, ..., X_n$. Here, we introduce infinitely many variables only for the convenience of formulating more formulas in a fixed alphabet.

### Markov Logic Networks

Let $\forall_{Y_1} ... \forall_{Y_n} \phi$ be the universal closure of a quantifier-free formula $\phi$, and let $\omega$ be a Herbrand structure. We find that $\forall_{Y_1} ... \forall_{Y_n} \phi$ is true in $\omega$ if all ground instances in $\bar{\phi}$ are true in $\omega$. Hence, the quantifiers $\forall_{Y_1} ... \forall_{Y_n}$ tell us how to aggregate the propositional truth values of the ground instances in $\bar{\phi}$ with respect to $\omega$ to assign a truth value to the formula $\forall_{Y_1} ... \forall_{Y_n} \phi$. A Markov logic network of Richardson and Domingos (46) now refines this aggregation procedure by introducing uncertainty. In particular, they replace these quantifiers with a real weight that indicates a degree of certainty.

**Definition 2.2.18** (Markov Logic Network)**.** A **Markov logic network** $\Phi$ is a finite set of **weighted constraints** $(w, \phi)$, where $w \in \mathbb{R} \cup \{\infty\}$ represents the weight of a quantifier-free formula $\phi$. For a ground substitution $\gamma : \{X_j\}_{j \in \mathbb{N}} \to \mathcal{C}$, we define the corresponding **ground instance** as $(w, \phi^\gamma)$. Finally, the **grounding** $\bar{\Phi}$ of the Markov logic network $\Phi$ consists of the ground instances of the weighted constraints in $\Phi$. The grounding of a Markov logic network forms a LogLinear model in the propositional alphabet $\mathfrak{P}(\mathfrak{A})$, which consists of all the ground atoms of the alphabet $\mathfrak{A}$. Consequently, the Markov logic network $\Phi$ induces a distribution $\pi$ on $\mathfrak{P}(\mathfrak{A})$-structures $\omega$, defined by

$$\pi(\omega) := \pi_\Phi(\omega) := \pi_{\bar{\Phi}}(\omega).$$

In particular, we first assign to every Herbrand structure $\omega$ the weight

$$w(\omega) := \exp\left( \sum_{\substack{(w,\phi) \in \Phi,\ w \neq \infty \\ n \text{ number of ground instances } \psi \in \bar{\phi} \\ \text{such that } \omega \models \psi}} n \cdot w \right) \tag{2.7}$$

whenever $\omega \models \bar{\phi}$ for all **hard constraints** $(\infty, \phi) \in \Phi$ and $w(\omega) = 0$ otherwise. Next, we define a normalizing constant

$$w(\top) := \sum_{\omega \text{ Herbrand structure}} w(\omega)$$

and assign to each Herbrand structure the probability

$$\pi(\omega) := \frac{w(\omega)}{w(\top)}. \tag{2.8}$$

**Remark 2.2.11.** Given a Herbrand structure $\omega$, the weights $w$ of the quantifier-free formulas $\phi$ in a Markov logic network $\Phi$ tell us that we aggregate the truth values of the ground instances in $\bar{\phi}$ according to Equations (2.7) and (2.8). In this sense, these weights generalize the quantifiers in a universally closed formula.

As for LogLinear models in Section 2.2.2, there is no intuitive description for weighted constraints $(w, \phi)$ with a real weight $w \in \mathbb{R}$ in a Markov logic network $\Phi$. However, if the weight $w = \infty$, all ground instances of $\phi$ are necessarily satisfied. In the context of first-order logic, this is equivalent to asserting the universal closure $\forall_{Y_1} ... \forall_{Y_n} \phi$ of the formula $\phi$.

**Example 2.2.24.** Consider two friends, Jakob and Kilian, denoted as *jakob* and *kilian*, who regularly meet in a pub. Furthermore, we denote by $smokes(jakob)$ and $smokes(kilian)$ the events that Jakob and Kilian smoke, respectively. We model this situation with the alphabet $\mathfrak{A}$, which consists of the constants $\mathcal{C} = \{jakob, kilian\}$ and the predicate *smokes* with arity one. Assuming a positive correlation between Jakob and Kilian smoking could result in the Markov logic network $\Phi$, which is defined by the weighted constraint:

$$(\ln(2), smokes(X) \wedge smokes(Y)).$$

The grounding $\bar{\Phi}$ of $\Phi$ is then given by the following weighted constraints.

$$\begin{array}{ll} (\ln(2), smokes(kilian)) & (\ln(2), smokes(kilian) \wedge smokes(jakob)) \\ (\ln(2), smokes(jakob)) & (\ln(2), smokes(jakob) \wedge smokes(kilian)) \end{array}$$

This is a LogLinear model in the propositional alphabet $\mathfrak{P}(\mathfrak{A}) := \{smokes(jakob), smokes(kilian)\}$. Hence, we have to consider the four $\mathfrak{P}(\mathfrak{A})$-structures with the following weights.

$$\begin{array}{ll} \omega_1 := \emptyset & w(\omega_1) := 1 \\ \omega_2 := \{smokes(jakob)\} & w(\omega_2) := 2 \\ \omega_3 := \{smokes(kilian)\} & w(\omega_3) := 2 \\ \omega_4 := \{smokes(jakob), smokes(kilian)\} & w(\omega_4) = 16. \end{array}$$

Overall, we find Jakob and Kilian smoking with probability

$$\pi(smokes(jakob)) = \pi(smokes(kilian)) = \frac{2+16}{21} = \frac{18}{21}.$$

**Example 2.2.25.** In the alphabet of Example 2.2.24, we can also consider the Markov logic network, consisting of the constraint $(\infty, \neg smokes(X))$, which means that nobody, i.e. neither Jakob nor Kilian is smoking.

### 2.2.4 Logic Programs and Probabilistic Logic Programs

As Decartes suggested, it is natural to express causal relationships in rules (32). Hence, it seems natural to model causal reasoning with logic programs, which themselves consist of such rules. Here, we introduce three well-established semantics for logic programs: the supported model semantics (25), the minimal (Herbrand) model semantics (59), as well as the stable model semantics (27). Next, we recall the abductive point of view on logic programming (22). Extending logic programs by degrees of certainty and proceeding as in the construction of Markov logic networks (46) leads to LP$^{\text{MLN}}$ programs (35). If, instead, we extend logic programs by independent Boolean random variables, we obtain ProbLog programs (21; 26). Finally, logic programs with annotated disjunctions (62) extend logic programming by probabilistic case distinctions. Before addressing general (probabilistic) logic programming, we investigate the propositional case.

#### Propositional Logic Programming

Logic programs consist of rules, also known as clauses. Fix a propositional alphabet $\mathfrak{P}$ to define these clauses; that is, the fragment of propositional logic in which we are interested.

**Definition 2.2.19** (Clause)**.** A **(normal) clause** $C$ is a formula of the form

$$(h \leftarrow (b_1 \wedge (b_2 \wedge (... \wedge b_n))...)),$$

which we also denote as

$$h \leftarrow b_1 \wedge ... \wedge b_n,\ h \leftarrow b_1, ..., b_n \text{ or } \operatorname{head}(C) \leftarrow \operatorname{body}(C),$$

Here, head$(C) := h \in \mathfrak{P}$ is an atom, referred to as the **head** of $C$ and body$(C) := \{b_1, ..., b_n\}$ is a finite set of literals, known as the **body** of $C$. If body$(C) = \emptyset$ is empty, we simply denote $C$ by $h$ and call $C$ a **fact**. Furthermore, we say that $C$ is **positive** if body$(C) \subseteq \mathfrak{P}$, i.e., it does not contain any negative literals.

**Example 2.2.26.** Recall the situation of Example 2.2.6, where we consider a road that passes along a field with a sprinkler in it. The sprinkler is activated, denoted *sprinkler* by a weather sensor whenever it is not cloudy, denoted $\neg$*cloudy*. In addition, the road pavement is wet, denoted *wet*, either when the sprinkler is on or when it rains, denoted *rain*. Lastly, when the road is wet, it becomes slippery, denoted *slippery*. In the propositional alphabet

$$\mathfrak{P} := \{cloudy, sprinkler, rain, wet, slippery\}$$

we can build the following clauses.

$$\begin{array}{ll} C_1 := & sprinkler \leftarrow \neg cloudy \\ C_2 := & wet \leftarrow rain, sprinkler \\ C_3 := & cloudy \end{array}$$

We observe that $C_3$ is a fact. Further, $C_2$ and $C_3$ are positive clauses, while $C_1$ is not a positive clause.

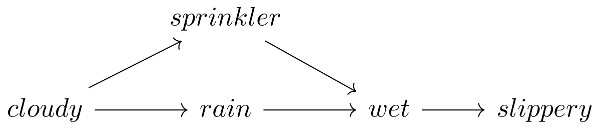
We proceed and construct programs from these clauses.

**Definition 2.2.20** (Logic Program, Dependence Graph)**.** A **logic program P** is a finite set of clauses. We call the program **P positive** if it consists only of positive clauses. The **dependence graph** graph(**P**) of **P** is the directed graph on the alphabet $\mathfrak{P}$ given by drawing an arrow $p \to q$ if and only if there exists a clause $C \in \mathbf{P}$ with head$(C) = q$ and body$(C) \cap \{p, \neg p\} \neq \emptyset$. A **cycle** is a finite alternating sequence of nodes and arrows of the form $q \to p_1 \to p_2 \to ... \to p_n \to q$ that ends and starts with the same node $q$. Finally, the program **P** is **acyclic** if its dependence graph graph(**P**) has no cycle.

**Example 2.2.27.** From the alphabet in Example 2.2.26, we can build the following positive program $\mathbf{P}_1$.

$$\begin{array}{ll} rain \leftarrow cloudy & \\ wet \leftarrow rain & wet \leftarrow sprinkler \\ slippery \leftarrow wet & \end{array}$$

Furthermore, by incorporating the clause $sprinkler \leftarrow \neg cloudy$ into $\mathbf{P}_1$, we obtain the program $\mathbf{P}_2$, which is no longer positive. We note that $\mathbf{P}_2$ is designed to model the scenario described in Example 2.2.26. Lastly, we obtain the dependence graph graph$(\mathbf{P}_2)$ of the program $\mathbf{P}_2$ below.



Hence, we conclude that $\mathbf{P}_2$ is an acyclic program

In logic programming, we consider a proposition $p$ to be true only if the program **P** forces $p$ to be true. Otherwise, according to the principle of **default negation**, the proposition $p$ is considered false. In particular, a negative literal $\neg p$ is true if there is no reason for the proposition $p$ to hold. This reflects that databases such as timetables usually state positive statements like departures of trains and not when nothing is departing.

**Example 2.2.28.** Recall the program $\mathbf{P}_2$ of Example 2.2.27. As there is no clause with the head *cloudy*, we conclude that *cloudy* cannot be enforced. Applying default negation, we find that *cloudy* is false, i.e. $\neg cloudy$ is true. Hence, we immediately obtain that *sprinkler*, *wet* and *slippery* need to be true. However, since the clause $rain \leftarrow cloudy$ is not applicable if *cloudy* is false, default negation yields that *rain* is false. In summary, the program $\mathbf{P}_2$ should have the unique model $\omega := \{sprinkler, wet, slippery\}$.

Subsequently, we recall three approaches to describe this intended behavior: the supported model semantics (25), minimal model semantics (59), and stable model semantics (27).

### The Supported Model Semantics of Acyclic Logic Programs

In the case of an acyclic program $\mathbf{P}$, Clark (16) observes that $\mathbf{P}$ can be translated to equivalent propositional formulas, stating that a valid proposition in a model $\omega$ needs to have a reason, i.e., a support in $\omega$.

**Definition 2.2.21** (Clark Completion, Supported Model Semantics for Logic Programs)**.** Let $\mathbf{P}$ be a logic program. The **Clark completion** of $\mathbf{P}$ is defined to be the set of formulas

$$\operatorname{comp}(\mathbf{P}) := \left\{ p \leftrightarrow \bigvee_{\substack{C \in \mathbf{P} \\ \operatorname{head}(C) = p}} \bigwedge \operatorname{body}(C) \right\}_{p \in \mathfrak{P}} .$$

Further, a model of the Clark completion $\omega \models \operatorname{comp}(\mathbf{P})$ is said to be a **supported model** of $\mathbf{P}$.

**Example 2.2.29.** The Clark completion of the program $\mathbf{P}_2$ in Example 2.2.27

$$\begin{array}{ll} rain \leftarrow cloudy & \\ sprinkler \leftarrow \neg cloudy & \\ wet \leftarrow rain & \qquad\qquad wet \leftarrow sprinkler \\ slippery \leftarrow wet & \end{array}$$

is given by

$$\begin{aligned} &cloudy \leftrightarrow \bot \\ &rain \leftrightarrow cloudy \\ &sprinkler \leftrightarrow \neg cloudy \\ &wet \leftrightarrow rain \lor sprinkler \\ &slippery \leftrightarrow wet \end{aligned}$$

As in Example 2.2.28, the program $\mathbf{P}_2$ has the supported model $\omega := \{sprinkler, wet, slippery\}$.

Although the supported model semantics is formally well-defined for general propositional logic programs, i.e., it associates a unique (possibly empty) set of models to each program $\mathbf{P}$, it is only guaranteed to represent the desired behavior of acyclic logic programs. In particular, the supported model semantics could yield counterintuitive results for cyclic programs.

**Example 2.2.30.** Assume $h_1$ and $h_2$ are two neighboring houses. In addition, let us denote by $fire(h_i)$ the event of a fire in the house $h_i$, $i = 1, 2$. It makes sense to assume that a fire in the house $h_1$ leads to a fire in the house $h_2$ and vice versa. So far, this situation can be modeled with the following program $\mathbf{P}$.

$$fire(h_2) \leftarrow fire(h_1) \qquad\qquad fire(h_1) \leftarrow fire(h_2)$$

Note that $\mathbf{P}$ has two supported models $\omega_1 := \emptyset$ and $\omega_2 := \{fire(h_2), fire(h_1)\}$. However, since we do not state a fact $fire(h_1)$ or $fire(h_2)$, we cannot enforce $fire(h_i)$, $i = 1, 2$ based on the program $\mathbf{P}$. Hence, $\omega_1 = \emptyset$ should be the only model of $\mathbf{P}$. This also reflects our intuition that houses do not spontaneously burn, just because they would potentially affect each other.

### The Minimal Model Semantics of Positive Logic Program

Recall that we identify $\mathfrak{P}$-structures $\omega \subseteq \mathfrak{P}$ with the set of all propositions $p \in \mathfrak{P}$ satisfied in $\omega$. Van Emden and Kowalski (59) propose the minimal model semantics for positive programs.

**Definition 2.2.22** (Minimal Model Semantics of Logic Programs)**.** A **minimal model** of a logic program $\mathbf{P}$ is an inclusion-minimal model $\omega$ of $\mathbf{P}$, where we consider $\mathbf{P}$ as a set of propositional formulas.

The immediate consequence operator allows us to compute minimal models of positive programs.

**Definition 2.2.23** (Immediate Consequence Operator)**.** Let $\mathbf{P}$ be a logic program. The **immediate consequence operator** $\mathcal{T} := \mathcal{T}_{\mathbf{P}} : \mathcal{P}(\mathfrak{P}) \to \mathcal{P}(\mathfrak{P})$ is defined by assigning to each structure $\omega \subseteq \mathfrak{P}$ the structure

$$\mathcal{T}(\omega) := \{\mathrm{head}(C) : C \in \mathbf{P}, \, \omega \models \mathrm{body}(C)\}.$$

**Example 2.2.31.** Recall the positive program $\mathbf{P}_1$ of Example 2.2.27

$$\begin{array}{ll} rain \leftarrow cloudy & \\ wet \leftarrow rain & wet \leftarrow sprinkler \\ slippery \leftarrow wet & \end{array}$$

and denote by $\mathcal{T}$ its immediate consequence operator. We observe that $\mathcal{T}(\{cloudy\}) = \{rain\}$.

**Lemma 2.2.1** (van Emden, Kowalski (59))**.** *Let $\boldsymbol{P}$ be a positive logic program. We recursively define the ascending sequence of structures $(\omega_i)_{i \in \mathbb{N}}$ by*

$$\omega_0 = \emptyset \qquad \textit{and} \qquad \omega_{i+1} := \mathcal{T}(\omega_i),$$

*where $\mathcal{T}$ denotes the immediate consequence operator. Since the alphabet $\mathfrak{P}$ is finite, there exists a structure $\omega$ and a natural number $n \in \mathbb{N}$ such that $\omega = \omega_i$ for all $i \geq n$. In this case, $\omega$ is the unique minimal model of the program $\boldsymbol{P}$.* □

**Example 2.2.32.** Consider the program $\mathbf{P}_1$ in Example 2.2.31. According to Lemma 2.2.1, the program $\mathbf{P}_1 \cup \{cloudy\}$ has the minimal model $\omega := \{cloudy, rain, wet, slippery\}$. As desired, we see that $\omega_1 := \emptyset$ is the unique minimal of the program $\mathbf{P}$ in Example 2.2.30.

$$\mathbf{P}: \qquad fire(h_2) \leftarrow fire(h_1) \qquad fire(h_1) \leftarrow fire(h_2)$$

Again, the minimal model semantics is formally well-defined for general propositional logic programs, i.e. it associates a unique (possibly empty) set of models to each program $\mathbf{P}$. However, it is only guaranteed to represent the desired behavior for positive logic programs. Specifically, as the following example from Gelfond and Lifschitz (27) demonstrates, the minimal model semantics could yield undesired results when considering clauses with negation in their body.

**Example 2.2.33.** Fix the alphabet $\mathfrak{P} := \{p(1,2), p(2,1), q(1), q(2)\}$ and consider the following logic program $\mathbf{P}$.

$$\begin{array}{l} p(1,2) \\ q(1) \leftarrow p(1,2), \neg q(2) \\ q(2) \leftarrow p(2,1), \neg q(1) \end{array}$$

As there is no clause with head $p(2,1)$, we intuitively assume by default negation that $p(2,1)$ is false. Therefore, the clause $q(2) \leftarrow p(2,1), \neg q(1)$ is not applicable and $\neg q(2)$ holds with the same argument. Hence, the clause $q(1) \leftarrow p(1,2), \neg q(2)$ enforces $q(1)$ to be true. We conclude that the program $\mathbf{P}$ should have the unique model $\omega_1 := \{q(1), p(1,2)\}$. However, the program $\mathbf{P}$ has two minimal models $\omega_1$ and $\omega_2 := \{q(2), p(1,2)\}$.

### The Stable Model Semantics of General Logic Programs

Finally, Gelfond and Lifschitz (27) propose the stable model semantics for general logic programs. In this thesis, we adopt the viewpoint that the stable model semantics formally defines the desired notion of model for general logic programs.

**Definition 2.2.24** (Stable Model Semantics of Logic Programs)**.** Let $\mathbf{P}$ be a logic program. Firstly, for a structure $\omega \subseteq \mathfrak{P}$, we construct the **reduct** given by the positive program

$$\mathbf{P}_\omega := \{\mathrm{head}(C) \leftarrow \mathrm{body}(C) \cap \mathfrak{P} : C \in \mathbf{P} \text{ such that } p \notin \omega \text{ whenever } \neg p \in \mathrm{body}(C) \setminus \mathfrak{P}\}\,.$$

The reduct $\mathbf{P}_\omega$ results from the program $\mathbf{P}$ by first erasing all clauses $C \in \mathbf{P}$ mentioning a negative body literal $\neg p \in \mathrm{body}(C)$ with $p \in \omega$ and by then deleting all negative literals from the remaining clauses. Further, a **stable model** $\omega$ of $\mathbf{P}$ is a model $\omega$ of $\mathbf{P}$ considered as a set of propositional formulas, which is also the unique minimal model of the reduct $\mathbf{P}_\omega$. Finally, a program $\mathbf{P}$ is called **consistent** if it has at least one stable model.

**Example 2.2.34.** As desired, the structure $\omega := \{\mathit{sprinkler}, \mathit{wet}, \mathit{slippery}\}$ is the unique stable model of the program $\mathbf{P}_2$ in Example 2.2.27.

$$\begin{array}{ll}
\mathit{rain} \leftarrow \mathit{cloudy} & \\
\mathit{sprinkler} \leftarrow \neg\mathit{cloudy} & \\
\mathit{wet} \leftarrow \mathit{rain} & \mathit{wet} \leftarrow \mathit{sprinkler} \\
\mathit{slippery} \leftarrow \mathit{wet} &
\end{array}$$

For the positive program in Example 2.2.30, we obtain the unique stable model $\omega_1 := \emptyset$.

$$\mathit{fire}(h_2) \leftarrow \mathit{fire}(h_1) \qquad\qquad \mathit{fire}(h_1) \leftarrow \mathit{fire}(h_2)$$

Finally, $\omega_1 := \{q(1), p(1,2)\}$ is the unique stable model of the program in Example 2.2.33.

$$\begin{array}{l}
p(1,2) \\
q(1) \leftarrow p(1,2), \neg q(2) \\
q(2) \leftarrow p(2,1), \neg q(1)
\end{array}$$

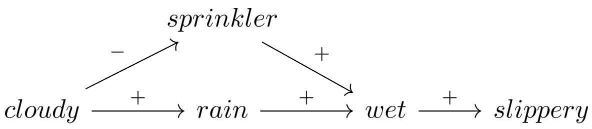
Assume that we want to decide whether a query $q$ is true or false with respect to the knowledge represented by a logic program $\mathbf{P}$. This is feasible only when the program $\mathbf{P}$ has a unique stable model. Now, we present a broad class of programs with unique stable models.

**Definition 2.2.25** (Signed Dependence Graph and Stratified Programs)**.** Let $\mathbf{P}$ be a logic program. An edge $p \overset{-}{\to} q$ in the dependence graph $\mathrm{graph}(\mathbf{P})$ is **negative** if there is a clause $C \in \mathbf{P}$ with $\mathit{head}(C) = q$ and $\neg p \in \mathrm{body}(C)$. If there is a clause $C \in \mathbf{P}$ with $\mathit{head}(C) = q$ and $p \in \mathrm{body}(C)$, the edge $p \overset{+}{\to} q$ is **positive**. Note that an edge can be both negative and positive simultaneously. If we add these edge labels to the dependence graph, we obtain the **signed dependence graph** $\mathrm{graph}^{\pm}(\mathbf{P})$. Finally, the program $\mathbf{P}$ is **stratified** if its signed dependence graph $\mathrm{graph}^{\pm}(\mathbf{P})$ does not contain a cycle with a negative edge.

**Example 2.2.35.** Recall the program $\mathbf{P}_2$ of Example 2.2.27 below.

$$\begin{array}{ll}
\mathit{rain} \leftarrow \mathit{cloudy} & \\
\mathit{sprinkler} \leftarrow \neg\mathit{cloudy} & \\
\mathit{wet} \leftarrow \mathit{rain} & \mathit{wet} \leftarrow \mathit{sprinkler} \\
\mathit{slippery} \leftarrow \mathit{wet} &
\end{array}$$

We obtain the following signed dependence graph $\mathrm{graph}^{\pm}(\mathbf{P}_2)$.

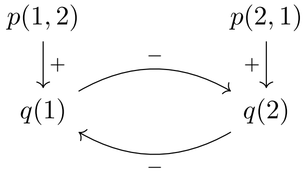


Hence, the program $\mathbf{P}_2$ is acyclic and stratified.

The program $\mathbf{P}$ of Example 2.2.33

$$\begin{aligned} &p(1,2) \\ &q(1) \leftarrow p(1,2), \neg q(2) \\ &q(2) \leftarrow p(2,1), \neg q(1) \end{aligned}$$

has the signed dependence graph $\mathrm{graph}^{\pm}(\mathbf{P})$ below.

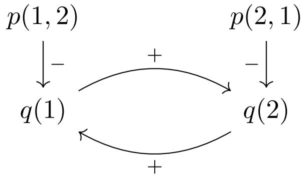


Hence, the program $\mathbf{P}$ is not stratified as we find the cycle $q(1) \overset{-}{\rightarrow} q(2) \overset{-}{\rightarrow} q(1)$.

Finally, consider the following program $\mathbf{P}'$.

$$\begin{aligned} &p(1,2) \\ &q(1) \leftarrow \neg p(1,2), q(2) \\ &q(2) \leftarrow \neg p(2,1), q(1) \end{aligned}$$

with the dependence graph



As the cycle $q(1) \overset{+}{\rightarrow} q(2) \overset{+}{\rightarrow} q(1)$ contains only positive edges, the program $\mathbf{P}'$ is stratified. It has the unique stable model $\{p(1,2)\}$.

As desired, every stratified program has a unique stable model.

**Theorem 2.2.2** (Gelfond and Lifschitz (27))**.** *Every stratified program has a unique stable model.* □

Finally, we find that every stable model is a model of the Clark completion.

**Theorem 2.2.3** (Gelfond and Lifschitz (27))**.** *Every stable model of a logic program is also a supported model* □*.*

**Corollary 2.2.4.** *If a consistent logic program* $\boldsymbol{P}$ *has a unique supported model* $\omega$*, we find that* $\omega$ *is also the unique stable model of* $\boldsymbol{P}$*. This is the case, for instance, when* $\boldsymbol{P}$ *is an acyclic program.* □

However, unique supported models or minimal models are not necessarily stable.

**Example 2.2.36.** Consider the program $\mathbf{P}$ that is given by

$$p \leftarrow \neg p \qquad\qquad p \leftarrow q \qquad\qquad q \leftarrow q.$$

Note that $\mathbf{P}$ has the unique supported model $\{p, q\}$, which is not stable. Observe that this does not contradict Corollary 2.2.4 as the program $\mathbf{P}$ is not consistent.

### Non-Ground Logic Programming

Let $\mathfrak{A}$ be a relational alphabet. In this case, a **(normal) clause** is a quantifier-free formula

$$(h \leftarrow (b_1 \wedge (b_2 \wedge (... \wedge b_n)...)))$$

that we denoted by $h \leftarrow b_1 \wedge ... \wedge b_n$, $h \leftarrow b_1, ..., b_n$ or $\mathrm{head}(C) \leftarrow \mathrm{body}(C)$. Here, $\mathrm{head}(C) := h$ is an atom and $\mathrm{body}(C) := \{b_1, ..., b_n\}$ is a finite set of literals. Furthermore, a **logic program** $\mathbf{P}$ again is a finite set of clauses and the **grounding** $\bar{\mathbf{P}}$ of $\mathbf{P}$ is the set of all ground instances of the clauses in $\mathbf{P}$. As the alphabet $\mathfrak{A}$ consists of finitely many constants and predicates, the grounding $\bar{\mathbf{P}}$ contains only finitely many clauses that do not mention any variables. Finally, a **minimal**, **stable** or **supported** model $\omega$ of $\mathbf{P}$ is a minimal, stable or supported model of the grounding $\bar{\mathbf{P}}$, treated as a propositional program in the ground atoms of the alphabet $\mathfrak{A}$.

**Example 2.2.37.** Consider the alphabet $\mathfrak{A} := (\mathcal{P}, \mathcal{C})$ consisting of the predicates

$$\mathcal{P} := \{smokes, spontaneously\_smokes\}$$

together with the constants

$$\mathcal{C} := \{jakob, kilian\}.$$

Each predicate has arity one, and the constants denote two friends, Jakob and Kilian, respectively. Furthermore, $spontaneously\_smokes(jakob)$ and $spontaneously\_smokes(kilian)$ indicate that Jakob and Kilian start to smoke, respectively. By $smokes(jakob)$ and $smokes(kilian)$, we mean that Jakob and Kilian are smoking, respectively. In the alphabet $\mathfrak{A}$, we can build the following program $\mathbf{P}$:

$$\begin{aligned}
&smokes(X) \leftarrow spontaneously\_smokes(X)\\
&smokes(Y) \leftarrow smokes(X)\\
&spontaneously\_smokes(kilian)
\end{aligned}$$

Hence, everyone who smokes spontaneously actually smokes. If one friend is smoking, so is the other friend. And Kilian starts smoking spontaneously. We obtain the grounding $\bar{\mathbf{P}}$ below:

$$\begin{aligned}
&smokes(kilian) \leftarrow spontaneously\_smokes(kilian)\\
&smokes(jakob) \leftarrow spontaneously\_smokes(jakob)\\
&smokes(kilian) \leftarrow smokes(kilian)\\
&smokes(jakob) \leftarrow smokes(kilian)\\
&smokes(jakob) \leftarrow smokes(jakob)\\
&smokes(kilian) \leftarrow smokes(jakob)\\
&spontaneously\_smokes(kilian)
\end{aligned}$$

Hence, we find that $\mathbf{P}$ has the unique minimal and stable model:

$$\omega := \{spontaneously\_smokes(kilian), smokes(kilian), smokes(jakob)\}.$$

Finally, note that $\omega \cup \{spontaneously\_smokes(jakob)\}$ would yield another supported model of $\mathbf{P}$.

### Propositional Abductive Logic Programming

Toward reasoning on Aristotelian knowledge, we introduce the abductive perspective on logic programming (22). The objective of the resulting abductive logic programs is to offer an explanation for a given set of observations. Once more, we start with the propositional case and fix a propositional alphabet $\mathfrak{P}$.

**Definition 2.2.26** (Abductive Logic Program)**.** An **integrity constraint** $IC$ is an expression of the form $\bot \leftarrow b_1 \wedge ... \wedge b_n$ also written $\bot \leftarrow \text{body}(IC)$, where $\text{body}(IC)$ is a finite set of literals. Further, an **abductive logic program** is a triplet $\mathcal{P} := (\mathbf{P}, \mathfrak{A}, \mathbf{IC})$ consisting of a logic program $\mathbf{P}$, a finite set of integrity constraints $\mathbf{IC}$ and a set of **abducibles** $\mathfrak{A} \subseteq \mathfrak{P}$ such that no abducible $u \in \mathfrak{A}$ is the head of a clause in $\mathbf{P}$. Finally, we say that $\mathcal{P}$ is **positive** if the underlying logic program $\mathbf{P}$ is.

In the context of databases, integrity constraints express a sanity check on data (15, Chapter 9). In this thesis, we use them to represent our observations, i.e., they ensure that the Aristotelian knowledge provided by the causal relations encoded in the program $\mathbf{P}$ and the explanations in $\mathfrak{A}$ is also consistent with the factual knowledge that we observe.

**Example 2.2.38.** Recall the situation of Example 2.2.6, where we consider a road that passes along a field with a sprinkler in it. The sprinkler is activated, denoted *sprinkler* by a weather sensor whenever it is not cloudy, denoted $\neg cloudy$. In addition, the road pavement is wet, denoted *wet*, either when the sprinkler is on or when it rains, denoted *rain*. Lastly, when the road is wet, it becomes slippery, denoted *slippery*. Consider the program $\mathbf{P}_2$ of Example 2.2.27

$$\begin{aligned} &rain \leftarrow cloudy && \\ &sprinkler \leftarrow \neg cloudy && \\ &wet \leftarrow rain && wet \leftarrow sprinkler \\ &slippery \leftarrow wet && \end{aligned}$$

and let *cloudy* be the only abducible, i.e. we find $\mathfrak{A} := \{cloudy\}$. We may also observe rainy weather, giving the integrity constraints $\mathbf{IC} := \{\bot \leftarrow \neg rain\}$. In this way, we obtain an abductive logic program $\mathcal{P} := (\mathbf{P}_2, \mathfrak{A}, \mathbf{IC})$.

Further, we recall the various semantics of an abductive logic program.

**Definition 2.2.27** (Model of an Abductive Logic Program)**.** A structure $\omega \subseteq \mathfrak{P}$ is a **minimal, stable**, or **supported model** of the abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{A}, \mathbf{IC})$ if $\omega$ satisfies the integrity constraints $\mathbf{IC}$, i.e., $\omega \models IC$ (meaning $\omega \not\models \text{body}(IC)$) for all $IC \in \mathbf{IC}$ and if $\omega$ is a minimal, stable, or supported model of the program $\mathbf{P} \cup (\omega \cap \mathfrak{A})$. We then call the set $\epsilon := \omega \cap \mathfrak{A}$ the **explanation** of $\omega$.

**Example 2.2.39.** In the situation of Example 2.2.38, the abductive logic program $\mathcal{P}' := (\mathbf{P}_2, \mathfrak{A}, \emptyset)$ has two stable models

$$\begin{aligned} \omega_1 &:= \{cloudy, rain, wet, slippery\} \\ \omega_2 &:= \{sprinkler, wet, slippery\} \end{aligned}$$

with explanations $\epsilon_1 := \{cloudy\}$ and $\epsilon_2 := \emptyset$, respectively. Since only $\omega_1$ is consistent with the observation of rainy weather expressed by the integrity constraint $\bot \leftarrow \neg rain$, we conclude that $\omega_1$ is the only stable model of the abductive logic program $\mathcal{P}$ in Example 2.2.38.

### Non-Ground Abductive Logic Programming

Let $\mathfrak{A}$ be a relational alphabet. In this case, an **integrity constraint** $IC$ is a quantifier-free formula $\bot \leftarrow b_1 \wedge ... \wedge b_n$ for literals $b_1$,...,$b_n$. Now an abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{Ab}, \mathbf{IC})$

is a tuple consisting of a logic program $\mathbf{P}$, a set of atoms $\mathfrak{Ab}$, called **abducibles**, and a set of integrity constraints $\mathbf{IC}$ such that the **grounding** $\bar{\mathcal{P}} := (\bar{\mathbf{P}}, \bar{\mathfrak{Ab}}, \bar{\mathbf{IC}})$ is an abductive logic program in the propositional alphabet $\mathfrak{P}(\mathfrak{A})$ given by the ground atoms of $\mathfrak{A}$. A **supported**, **minimal** or **stable model** $\omega$ of $\mathcal{P}$ with **explanation** $\epsilon$ then is a supported, minimal or stable model $\omega$ of the grounding $\bar{\mathcal{P}}$ with explanation $\epsilon$.

**Example 2.2.40.** Recall the alphabet $\mathfrak{A} := (\mathcal{P}, \mathcal{C})$ of Example 2.2.37, where

$$\mathcal{P} := \{\mathit{smokes}, \mathit{spontaneously\_smokes}\} \text{ and } \mathcal{C} := \{\mathit{jakob}, \mathit{kilian}\}.$$

Each predicate has arity one, and the constants denote two friends, Jakob and Kilian, respectively. Furthermore, $\mathit{spontaneously\_smokes}(\mathit{jakob})$ and $\mathit{spontaneously\_smokes}(\mathit{kilian})$ indicate that Jakob and Kilian start to smoke, respectively. By $\mathit{smokes}(\mathit{jakob})$ and $\mathit{smokes}(\mathit{kilian})$, we mean that Jakob and Kilian are smoking, respectively.

We can now construct the abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{Ab}, \mathbf{IC})$, consisting of the logic program $\mathbf{P}$ given by

$$\mathit{smokes}(X) \leftarrow \mathit{spontanously\_smokes}(X) \qquad \mathit{smokes}(Y) \leftarrow \mathit{smokes}(X)$$

the abducibles $\mathfrak{Ab} := \{\mathit{spontanously\_smokes}(X)\}$ and the constraints

$$\mathbf{IC} := \{\bot \leftarrow \mathit{spontanously\_smokes}(\mathit{jakob})\}.$$

In this case, we find that $\mathcal{P}$ has the stable models

$$\omega_1 := \{\mathit{spontanously\_smokes}(\mathit{kilian}), \mathit{smokes}(\mathit{kilian}), \mathit{smokes}(\mathit{jakob})\}$$
$$\omega_2 := \emptyset$$

with explanations $\epsilon_1 := \{\mathit{spontanously\_smokes}(\mathit{kilian})\}$ and $\epsilon_2 := \emptyset$, respectively.

### Probabilistic Logic Programming

To conclude, we delve into the realm of probabilistic extensions of logic programming provided by the distribution semantics of Sato (55) and Poole (43). Here, we fix a propositional alphabet $\mathfrak{P}$ and introduce the various concepts in probabilistic logic programming for the propositional case.

Generally, in all of these concepts, one chooses clauses in a given logic program $\mathbf{P}$ at random, yielding a distribution on the resulting programs $\mathbf{P}' \subseteq \mathbf{P}$. In addition, we apply the P-log semantics of Baral et al. (6) and extend this distribution on logic programs to a distribution on the corresponding stable models (27). The presented languages now differ in how they state the random choice of clauses in the program $\mathbf{P}$.

### Logic Programs with Annotated Disjunctions

Vennekens et al. (60) introduce a causal semantics for logic programs with annotated disjunctions that also enables counterfactual reasoning (61). We compare their causal reasoning with our approach in Section 3.4.4. Generally, in logic programs with annotated disjunctions, Vennekens and Verbaeten (62) express random choices of clauses with disjunctions in the head, where every disjunct is labeled with a probability.

**Example 2.2.41.** Consider a road that passes through a field with a sprinkler in it. Assume it is cloudy, denoted *cloudy*, with probability 0.5. Further, if it is cloudy, it rains, denoted *rain*, with probability 0.6. The sprinkler is then switched on, denoted *sprinkler*, by a weather sensor with probability 0.1 if it is cloudy and it does not rain, denoted $\neg \mathit{rain}$, and with probability 0.8 if it is sunny. The road pavement is wet, denoted *wet*, with probability 0.7 if it rains and probability 0.3

if the sprinkler is on. Finally, a wet road is slippery, denoted slippery, with a probability of 0.9. We model this situation in the following logic program with annotated disjunctions $\mathbf{P}$:

$$\begin{aligned}
&C_1 := \mathit{cloudy} : 0.5\\
&C_2 := \mathit{rain} : 0.6; \mathit{sprinkler} : 0.1 \leftarrow \mathit{cloudy}\\
&C_3 := \mathit{sprinkler} : 0.8 \leftarrow \neg\mathit{cloudy}\\
&C_4 := \mathit{wet} : 0.7 \leftarrow \mathit{rain}\\
&C_5 := \mathit{wet} : 0.3 \leftarrow \mathit{sprinkler}\\
&C_6 := \mathit{slippery} : 0.9 \leftarrow \mathit{wet}
\end{aligned}$$

Here, clause $C_2$, for instance, means that we choose the clause *rain* ← *cloudy* with probability 0.6 and the clause *sprinkler* ← *cloudy* with probability 0.1.

Recall the syntax of logic programs with annotated disjunctions.

**Definition 2.2.28** (Logic Program with Annotated Disjunction)**.** We call an expression

$$C := h_1 : \pi_1; ...; h_l : \pi_l \leftarrow b_1, ..., b_n$$

a **clause with annotated disjunctions** or **LPAD clause** if the assertions below are satisfied.

i) We have that $\text{head}(C) := (h_1, ..., h_l)$ is a tuple of propositions called the **head** of $C$. We write $h \in (h_1, ..., h_l)$ if $h = h_i$ for $1 \leq i \leq l$. Further, we write $l(C) := l$ and $h_i(C) := h_i$ for $1 \leq i \leq l$.

ii) We have that $\text{body}(C) := \{b_1, ..., b_n\}$ is a finite set of literals called the **body** of $C$.

iii) For all $1 \leq i \leq l$, the **probability** of the head atom $h_i$ is given by a number $\pi_i(C) := \pi_i \in [0, 1]$, such that $\sum_i \pi_i \leq 1$.

Informally, the clause $C$ means that we independently choose the heads $h_1$ or $h_2$ or... or $h_l$ for the clause $h_i \leftarrow b_1, ..., b_n$ with probability $\pi_i$.

A **logic program with annotated disjunctions** or **LPAD** is then a finite set of LPAD clauses $\mathbf{P}$. A **selection** of an LPAD $\mathbf{P}$ is a function $\sigma : \mathbf{P} \to \mathbb{N} \cup \{\bot\}$ that assigns to each LPAD clause $C \in \mathbf{P}$ a natural number $1 \leq \sigma(C) \leq l(C)$ or the symbol $\sigma(C) :=\bot \notin \mathbb{N}$. Assuming that all choices of heads are mutually independent, to each selection $\sigma$ we associate the probability

$$\pi(\sigma) := \prod_{\substack{C\in\mathbf{P}\\ \sigma(C)\in\mathbb{N}}} \pi_{\sigma(C)}(C) \cdot \prod_{\substack{C\in\mathbf{P}\\ \sigma(C)=\bot}} \left(1 - \sum_{i=1}^{l(C)} \pi_i(C)\right)$$

and the logic program

$$\mathbf{P}^\sigma := \left\{h_{\sigma(C)} \leftarrow \text{body}(C) : C \in \mathbf{P},\ \sigma(C) \neq\bot\right\}.$$

We call a selection $\sigma$ **consistent** if the logic program $\mathbf{P}^\sigma$ is consistent, that is, $\mathbf{P}^\sigma$ has at least one stable model. The event consistent($\mathbf{P}$) that the LPAD $\mathbf{P}$ is **consistent** is then given by the set of all consistent selections, i.e.

$$\text{consistent}(\mathbf{P}) := \{\sigma \text{ consistent selection of the LPAD } \mathbf{P}\}.$$

Using the principle of indifference in Law 3, we extend the probability distribution on the selections $\sigma$, that is on the logic programs $\mathbf{P}^\sigma$, to a distribution on the stable models $\omega$ of these programs $\mathbf{P}^\sigma$:

Let $\sigma$ be a selection. If $\sigma$ is not consistent, there is no stable model of $\mathbf{P}^\sigma$ and we conclude that $\sigma$ cannot be observed. Hence, we observe that the LPAD $\mathbf{P}$ is consistent, that is, we observe the event consistent($\mathbf{P}$). Assuming that $\mathbf{P}^\sigma$ has the stable models $\omega_1(\sigma), ..., \omega_{n(\sigma)}(\sigma)$, $n(\sigma) \in \mathbb{N}$ for every consistent selection $\sigma$, the **P-log semantics** assigns to every world $\omega$ the probability

$$\pi(\omega) := \pi_{\mathbf{P}}(\omega) := \pi_{\mathbf{P}}^{P-log}(\omega) := \sum_{\substack{\sigma \text{ consistent selection} \\ 1 \leq i \leq n(\sigma) \\ \omega = \omega_i(\sigma)}} \frac{\pi(\sigma | \operatorname{consistent}(\mathbf{P}))}{n(\sigma)} =$$

$$= \frac{1}{\pi(\operatorname{consistent}(\mathbf{P}))} \sum_{\substack{\sigma \text{ consistent selection} \\ 1 \leq i \leq n(\sigma) \\ \omega = \omega_i(\sigma)}} \frac{\pi(\sigma)}{n(\sigma)}.$$

Once again, to a formula $\phi$ we associate the probability

$$\pi(\phi) := \pi_{\mathbf{P}}(\phi) := \pi_{\mathbf{P}}^{P-log}(\phi) := \sum_{\substack{\omega \text{ world} \\ \omega \models \phi}} \pi(\omega).$$

**Example 2.2.42.** To enforce the proposition *rain* in Example 2.2.41, we need to choose the clauses *cloudy* and *rain* ← *cloudy*, that is, it is raining with probability

$$\pi(rain) := 0.5 \cdot 0.6 = 0.3.$$

Lastly, let $\mathfrak{A}$ be a relational alphabet. We call an expression of the form

$$C := h_1 : \pi_1; ...; h_l : \pi_l \leftarrow b_1, ..., b_n$$

an **LPAD clause** if the following assertions are satisfied:

i) We have that head($C$) := $(h_1, ..., h_l)$ is a tuple of atoms called the **head** of $C$.

ii) We have that body($C$) := $\{b_1, ..., b_n\}$ is a finite set of literals called the **body** of $C$.

iii) For all $1 \leq i \leq l$ the **probability** of the head atom $h_i$ is given by a number $\pi_i(C) := \pi_i \in [0, 1]$, such that $\sum_i \pi_i \leq 1$.

For a ground substitution $\gamma$ the corresponding ground instance is defined by

$$C^\gamma := h_1^\gamma : \pi_1; ...; h_l^\gamma : \pi_l \leftarrow b_1^\gamma, ..., b_n^\gamma.$$

Again, an **LPAD P** is a finite set of LPAD clauses and its **grounding $\bar{\mathbf{P}}$** is the set of the corresponding ground instances, which is an LPAD in the propositional alphabet $\mathfrak{P}(\mathfrak{A})$ of all ground atoms of $\mathfrak{A}$. Hence, we define the probability of a Herbrand structure $\omega$ by grounding and set $\pi(\omega) := \pi_{\mathbf{P}}(\omega) := \pi_{\bar{\mathbf{P}}}(\omega)$.

### ProbLog – a Probabilistic Prolog

In this thesis, we mainly formulate our causal reasoning in the language of ProbLog. De Raedt et al. (21) and Fierens et al. (26) introduce ProbLog as the simplest probabilistic extension of Prolog. The main idea here is that it is not necessary to describe the choice of arbitrary clauses in the distribution semantics. Instead, it is sufficient if one chooses facts at random.

**Example 2.2.43.** Let us drop the mutual exclusivity of *rain* and *sprinkler* in Example 2.2.41. Hence, assume that it is cloudy, denoted *cloudy*, with probability 0.5. Further, if it is cloudy, it rains, denoted *rain*, with probability 0.6. The sprinkler is then switched on, denoted *sprinkler*, by a weather sensor with probability 0.1 if it is cloudy and probability 0.7 if it is sunny. The pavement of the road is wet, denoted *wet*, with probability 0.8 if it rains and probability 0.3 if the sprinkler is on. Finally, a wet road is slippery, denoted *slippery*, with a probability of 0.9. We model this situation in the following ProbLog program $\mathbf{P}$:

**Random facts** Facts($\mathbf{P}$)**:**

$0.5 :: u_1$ $0.6 :: u_2$ $0.7 :: u_3$ $0.1 :: u_4$ $0.8 :: u_5$ $0.3 :: u_6$ $0.9 :: u_7$

**Underlying Logic Program** LP($\mathbf{P}$) :

$\mathit{cloudy} \leftarrow u_1$

$\mathit{rain} \leftarrow \mathit{cloudy}, u_2$

$\mathit{sprinkler} \leftarrow \neg \mathit{cloudy}, u_3$ $\quad$ $\mathit{sprinkler} \leftarrow \mathit{cloudy}, u_4$

$\mathit{wet} \leftarrow \mathit{rain}, u_5$ $\quad$ $\mathit{wet} \leftarrow \mathit{sprinkler}, u_6$

$\mathit{slippery} \leftarrow \mathit{wet}, u_7$

Here, the random fact $0.6 :: u_2$ indicates that we independently choose the fact $u_2$ to be true with probability 0.6. These random facts then serve as switches in an underlying logic program, that is, in our case the choice of $u_2$ means that we choose the clause $\mathit{rain} \leftarrow \mathit{cloudy}$ with a probability of 0.6.

Let us now formally introduce the syntax of ProbLog programs.

**Definition 2.2.29** (Random Fact, ProbLog Program)**.** A **random fact** $RF$ is an expression

$$\pi(RF) :: u(RF),$$

where $u(RF) \notin \mathfrak{P}$ is the **error term** of $RF$ and where $\pi(RF) \in [0,1]$ is the **probability** of $u(RF)$. It indicates that we choose $u(RF)$ independently to be true with probability $\pi(RF)$.

A **ProbLog program P** consists of a finite set of random facts Facts($\mathbf{P}$) and an **underlying logic program** LP($\mathbf{P}$) in the alphabet $\mathfrak{P} \cup \{u(RF) : RF \in \mathrm{Facts}(\mathbf{P})\}$ such that no error term is the head of a clause in LP($\mathbf{P}$).

**Remark 2.2.12.** Let $\mathbf{P}$ be a ProbLog program and denote by $\mathfrak{A}$ the set of error terms. In this case, we find that $\mathcal{P} := (\mathrm{LP}(\mathbf{P}), \mathfrak{A}, \emptyset)$ is an abductive logic program without integrity constraints. This observation inspires our causal interpretation of ProbLog in Section 3.4.3.

From the independence of the error terms and the principle of indifference in Law 3 we obtain the P-log semantics of Baral et al. (6).

**Definition 2.2.30** (P-log Semantics of ProbLog Programs)**.** Let $\mathbf{P}$ be a ProbLog program. A **choice** $\epsilon$ is a subset of error terms. Interpreting error terms $u$ as mutually independent Boolean random variables that hold with the probabilities $\pi$ in the random facts $\pi :: u$, we assign to each choice $\epsilon$ the probability

$$\pi(\epsilon) := \prod_{\substack{RF \in \mathrm{Facts}(\mathbf{P}) \\ u(RF) \in \epsilon}} \pi(RF) \cdot \prod_{\substack{RF \in \mathrm{Facts}(\mathbf{P}) \\ u(RF) \notin \epsilon}} (1 - \pi(RF)).$$

We call a choice $\epsilon$ **consistent** if the logic program $\mathrm{LP}(\mathbf{P}) \cup \epsilon$ is consistent, that is, $\mathrm{LP}(\mathbf{P}) \cup \epsilon$ has at least one stable model. The event consistent($\mathbf{P}$) of the ProbLog program $\mathbf{P}$ being **consistent** is then given by the set of all consistent choices, i.e.

$$\mathrm{consistent}(\mathbf{P}) := \{\epsilon \text{ consistent choice of the ProbLog program } \mathbf{P}\}.$$

Using the principle of indifference in Law 3, we extend the probability distribution on the choices $\epsilon$, that is on the logic programs $\mathrm{LP}(\mathbf{P}) \cup \epsilon$, to a distribution on the stable models $\omega$ of these logic programs $\mathrm{LP}(\mathbf{P}) \cup \epsilon$:

Let $\epsilon$ be a choice. If $\epsilon$ is not consistent, there is no stable model of $\mathrm{LP}(\mathbf{P}) \cup \epsilon$ and we conclude that $\epsilon$ cannot be observed. Hence, we observe that the ProbLog program $\mathbf{P}$ is consistent, that is, we observe the event consistent($\mathbf{P}$). Assume that $\epsilon$ is a consistent choice and that the logic program $\mathrm{LP}(\mathbf{P}) \cup \epsilon$ has $n \in \mathbb{N}$ stable models $\omega_1, ..., \omega_n$. The principle of indifference in Law 3 then yields the **P-log semantics**

$$\pi(\omega_i) := \pi_{\mathbf{P}}(\omega_i) := \pi_{\mathbf{P}}^{P-log}(\omega_i) := \frac{\pi(\epsilon | \, \mathrm{consistent}(\mathbf{P}))}{n} =$$
$$= \frac{\pi(\epsilon)}{n \cdot \pi(\mathrm{consistent}(\mathbf{P}))} \text{ for all } 1 \leq i \leq n.$$

Setting $\pi(\omega) := \pi_{\mathbf{P}}(\omega) := \pi_{\mathbf{P}}^{P-log}(\omega) := 0$ for all structures $\omega$ that are not a stable model of $\mathrm{LP}(\mathbf{P}) \cup \epsilon$ for a choice $\epsilon$, we obtain a probability distribution on the $\mathfrak{P}$-structures. Finally, we define the probability of a formula $\phi$ to be true by

$$\pi(\phi) = \pi_{\mathbf{P}}^{P-log}(\phi) := \sum_{\substack{\omega \text{ structure} \\ \omega \models \phi}} \pi_{\mathbf{P}}(\omega).$$

**Remark 2.2.13.** We use the letter $\epsilon$ to denote the choices of a ProbLog program since they also denote explanations of the associated abductive logic program in Remark 2.2.12.

**Example 2.2.44.** Recall the ProbLog program of Example 2.2.43.

**Random facts** Facts($\mathbf{P}$)**:**

$0.5 :: u_1$ $0.6 :: u_2$ $0.7 :: u_3$ $0.1 :: u_4$ $0.8 :: u_5$ $0.3 :: u_6$ $0.9 :: u_7$

**Underlying Logic Program** LP($\mathbf{P}$) :

$\mathit{cloudy} \leftarrow u_1$

$\mathit{rain} \leftarrow \mathit{cloudy}, u_2$

$\mathit{sprinkler} \leftarrow \neg \mathit{cloudy}, u_3$ $\quad$ $\mathit{sprinkler} \leftarrow \mathit{cloudy}, u_4$

$\mathit{wet} \leftarrow \mathit{rain}, u_5$ $\quad$ $\mathit{wet} \leftarrow \mathit{sprinkler}, u_6$

$\mathit{slippery} \leftarrow \mathit{wet}, u_7$

It rains only if we choose $u_1$ and $u_2$ to be true, i.e., we find rainy weather with a probability of

$$\pi(\mathit{rain}) := \pi(u_1) \cdot \pi(u_2) = 0.5 \cdot 0.6 = 0.3.$$

**Example 2.2.45.** Assume that Kailin and Felix drink tea and the bell rings. If Kailin or Felix hear the bell, one of them stands up to open the door, while the other remains seated. This can be modeled with the following ProbLog program $\mathbf{P}$.

**Random facts** Facts($\mathbf{P}$)**:**

$0.5 :: u_1$ $\quad$ $0.5 :: u_2$

**Underlying Logic Program** LP($\mathbf{P}$) :

$felix \leftarrow \neg kailin, u_1$

$kailin \leftarrow \neg felix, u_2$

In this case, we obtain the following choices:

$$\epsilon_1 := \emptyset \qquad \epsilon_2 := \{u_1\}$$
$$\epsilon_3 := \{u_2\} \qquad \epsilon_4 := \{u_1, u_2\}$$

All choices $\epsilon_i$ have the same probability $\pi(\epsilon_i) = 1/4$. The program $\mathrm{LP}(\mathbf{P}) \cup \epsilon_1$ has the stable model $\omega_1 := \emptyset$, the program $\mathrm{LP}(\mathbf{P}) \cup \epsilon_2$ has the stable model $\omega_2 := \{u_1, felix\}$, and the program $\mathrm{LP}(\mathbf{P}) \cup \epsilon_3$ has the stable model $\omega_3 := \{u_2, kailin\}$. Finally, the program $\mathrm{LP}(\mathbf{P}) \cup \epsilon_4$ has stable models $\omega_4 := \{u_1, u_2, kailin\}$ and $\omega_5 := \{u_1, u_2, felix\}$. We find $\pi(\omega_1) = \pi(\omega_2) = \pi(\omega_3) = 1/4$. To assign a probability to $\omega_4$ and $\omega_5$, we apply the principle of indifference; since we do not have information distinguishing these worlds, we conclude that they are equally likely, i.e., we obtain $\pi(\omega_4) = \pi(\omega_5) = 1/8$. In this way, we find that Felix opens the door with a probability of $\pi(felix) = 3/8$.

Let us add the clause $felix \leftarrow \neg felix$, which models the constraint that Felix opens the door. Hence, we consider the following ProbLog program $\mathbf{P}'$:

**Random facts** $\mathrm{Facts}(\mathbf{P}')$**:**

$0.5 :: u_1$ $\qquad$ $0.5 :: u_2$

**Underlying Logic Program** $\mathrm{LP}(\mathbf{P}')$ :

$felix \leftarrow \neg kailin, u_1$

$kailin \leftarrow \neg felix, u_2$

$felix \leftarrow \neg felix$

The programs $\mathrm{LP}(\mathbf{P}') \cup \epsilon_1$ and $\mathrm{LP}(\mathbf{P}') \cup \epsilon_3$ have no stable model, i.e., the choices $\epsilon_1$ and $\epsilon_3$ are not consistent. The program $\mathrm{LP}(\mathbf{P}') \cup \epsilon_2$ has the stable model $\omega_2 = \{u_1, felix\}$ and the program $\mathrm{LP}(\mathbf{P}') \cup \epsilon_4$ has the stable model $\omega_5 = \{u_1, u_2, felix\}$, i.e., the choices $\epsilon_2$ and $\epsilon_4$ are consistent and the event of the ProbLog program $\mathbf{P}'$ being consistent is given by

$$\mathrm{consistent}(\mathbf{P}') = \{\epsilon_2, \epsilon_4\}.$$

If we naively set $\pi(\omega_2) = \pi(\omega_5) = 1/4$, the probability mass would not sum up to one, i.e. we normalize and divide by $\pi(\mathrm{consistent}(\mathbf{P}')) = 1/2$, which is the same as conditioning on $\mathrm{consistent}(\mathbf{P}')$. In other words, when extending the distribution of choices $\epsilon_1 - \epsilon_4$ to the stable models $\omega_2$ and $\omega_5$ of the programs $\mathrm{LP}(\mathbf{P}') \cup \epsilon_1 - \mathrm{LP}(\mathbf{P}') \cup \epsilon_4$, we can observe consistent choices. In summary, we find $\pi(\omega_2) = \pi(\omega_5) = 1/2$ and Felix opens the door with probability $\pi(felix) = 1$.

A logic program with annotated disjunctions $\mathbf{P}$ can be translated into a ProbLog program $\mathrm{Prob}(\mathbf{P})$ with the same P-log semantics.

**Definition 2.2.31** (De Raedt et al. (20))**.** Let $\mathbf{P}$ be an LPAD in $\mathfrak{P}$ and choose for every LPAD clause $C \in \mathbf{P}$ and for every natural number $1 \leq i \leq l(C)$ distinct propositions $h_i^C, u_i(C) \notin \mathfrak{P}$. The **ProbLog transformation** $\mathrm{Prob}(\mathbf{P})$ of the LPAD $\mathbf{P}$ is the ProbLog program that is given by the logic program $\mathrm{LP}(\mathrm{Prob}(\mathbf{P}))$, which consists of the clauses

$$h_i^C \leftarrow \mathrm{body}(C) \cup \{\neg h_j^C | 1 \leq j < i\} \cup \{u_i(C)\}$$
$$h_i \leftarrow h_i^C$$

for every LPAD clause $C \in \mathbf{P}$ and for every $1 \leq i \leq l(C)$ as well as the random facts

$$\mathrm{Facts}(\mathrm{Prob}(\mathbf{P})) := \left\{ \frac{\pi_i(C)}{1 - \prod_{1 \leq j < i} \pi_j(C)} :: u_i(C) \mid C \in \mathbf{P},\ 1 \leq i \leq l(C) \right\}.$$

In fact, we obtain the following result.

**Theorem 2.2.5** (De Raedt et al. (20))**.** *Let* $\boldsymbol{P}$ *be an LPAD. In this case, for every selection* $\sigma$ *of* $\boldsymbol{P}$*, we obtain a set of choices* $\epsilon(\sigma)$ *of the ProbLog program* $\mathrm{Prob}(\boldsymbol{P})$*. Here,* $\epsilon \in \epsilon(\sigma)$ *whenever* $u_i(C) \notin \epsilon$ *if* $\sigma(C) =\perp$*, or if* $i > \sigma(C)$ *and* $u_i(C) \in \epsilon$ *if* $\sigma(C) = i$*.*

*We observe that* $\boldsymbol{P}^\sigma$ *yields the same stable models as the logic programs* $\mathrm{LP}(\mathrm{Prob}(\boldsymbol{P})) \cup \epsilon$ *for every* $\epsilon \in \epsilon(\sigma)$*, and that* $\pi(\epsilon(\sigma)) = \pi(\sigma)$*. Furthermore, the LPAD* $\boldsymbol{P}$ *and its ProbLog transformation* $\mathrm{Prob}(\boldsymbol{P})$ *yield the same probability for every world* $\omega$*, i.e.,* $\pi_{\boldsymbol{P}}(\omega) = \pi_{\mathrm{Prob}(\boldsymbol{P})}(\omega)$*.* □

**Example 2.2.46.** Recall the logic program with annotated disjunction **P** from Example 2.2.41.

$$\begin{array}{l}
\mathit{cloudy} : 0.5 \\
\mathit{rain} : 0.6; \mathit{sprinkler} : 0.1 \leftarrow \mathit{cloudy} \\
\mathit{sprinkler} : 0.8 \leftarrow \neg \mathit{cloudy} \\
\mathit{wet} : 0.7 \leftarrow \mathit{rain} \\
\mathit{wet} : 0.3 \leftarrow \mathit{sprinkler} \\
\mathit{slippery} : 0.9 \leftarrow \mathit{wet}
\end{array}$$

The program **P** then gives rise to the following ProbLog transformation Prob(**P**).

**Random facts:**

$0.5 :: u_1(C_1)$ $0.6 :: u_1(C_2)$ $0.25 :: u_2(C_2)$ $0.8 :: u_1(C_3)$ $0.7 :: u_1(C_4)$ $0.3 :: u_1(C_5)$ $0.9 :: u_1(C_6)$

**Underlying Logic Program:**

$$\begin{array}{ll}
\mathit{cloudy}^{C_1} \leftarrow u_1(C_1) & \mathit{cloudy} \leftarrow \mathit{cloudy}^{C_1} \\
\mathit{rain}^{C_2} \leftarrow \mathit{cloudy}, u_1(C_2) & \mathit{rain} \leftarrow \mathit{rain}^{C_2} \\
\mathit{sprinkler}^{C_2} \leftarrow \mathit{cloudy}, \neg \mathit{rain}^{C_2}, u_2(C_2) & \mathit{sprinkler} \leftarrow \mathit{sprinkler}^{C_2} \\
\mathit{sprinkler}^{C_3} \leftarrow \neg \mathit{cloudy}, u_1(C_3) & \mathit{sprinkler} \leftarrow \mathit{sprinkler}^{C_3} \\
\mathit{wet}^{C_4} \leftarrow \mathit{rain}, u_1(C_4) & \mathit{wet} \leftarrow \mathit{wet}^{C_4} \\
\mathit{wet}^{C_5} \leftarrow \mathit{sprinkler}, u_1(C_5) & \mathit{wet} \leftarrow \mathit{wet}^{C_5} \\
\mathit{slippery}^{C_6} \leftarrow \mathit{wet}, u_1(C_6) & \mathit{slippery} \leftarrow \mathit{slippery}^{C_6}
\end{array}$$

Finally, each ProbLog program can be interpreted as an LPAD using the following mapping.

**Definition 2.2.32** (Riguzzi (47, §2.4))**.** Let **P** be a ProbLog program. In this case, we define the **LPAD transformation** LPAD(**P**) of **P** to be the LPAD containing one clause of the form $u(RF) : \pi(RF) \leftarrow$ for every random fact $\pi(RF) :: u(RF)$ in **P** and a clause of the form $\mathit{head}(C) : 1 \leftarrow \mathrm{body}(C)$ for every clause $C \in \mathrm{LP}(\mathbf{P})$. Furthermore, each selection $\sigma$ of LPAD(**P**) with a non-zero probability corresponds to a unique choice $\epsilon(\sigma)$ such that $u(RF) \in \epsilon(\sigma)$ if and only if $\sigma(u(RF) : \pi(RF) \leftarrow) \neq \perp$.

Once again, we observe that the LPAD transformation preserves the P-log semantics.

**Theorem 2.2.6** (Riguzzi (47), §2.4)**.** *In Definition 2.2.32, we observe that* $\mathrm{LP}(\boldsymbol{P}) \cup \epsilon(\sigma)$ *and* $\mathrm{LPAD}(\boldsymbol{P})^{\sigma}$ *produce the same stable models, and* $\pi(\sigma) = \pi(\epsilon(\sigma))$. *Consequently, the programs* ***P*** *and* LPAD(***P***) *assign the same probability to every world* $\omega$, *i.e.,* $\pi_{\boldsymbol{P}}(\omega) = \pi_{\mathrm{LPAD}(\boldsymbol{P})}(\omega)$. □

**Example 2.2.47.** Recall the ProbLog program of Example 2.2.43.

**Random facts** Facts(**P**)**:**

$0.5 :: u_1$ $0.6 :: u_2$ $0.7 :: u_3$ $0.1 :: u_4$ $0.8 :: u_5$ $0.3 :: u_6$ $0.9 :: u_7$

**Underlying Logic Program** LP(**P**) :

$$\begin{array}{ll}
\mathit{cloudy} \leftarrow u_1 & \\
\mathit{rain} \leftarrow \mathit{cloudy}, u_2 & \\
\mathit{sprinkler} \leftarrow \neg \mathit{cloudy}, u_3 & \mathit{sprinkler} \leftarrow \mathit{cloudy}, u_4 \\
\mathit{wet} \leftarrow \mathit{rain}, u_5 & \mathit{wet} \leftarrow \mathit{sprinkler}, u_6 \\
\mathit{slippery} \leftarrow \mathit{wet}, u_7 &
\end{array}$$

The LPAD transformation of the program **P** results in the following LPAD LPAD(**P**).

$$\begin{aligned}
&u_1 : 0.5\ u_2 : 0.6\ u_3 : 0.8\ u_4 : 0.1\ u_5 : 0.7\ u_6 : 0.3\ u_7 : 0.9\\
&\mathit{cloudy} : 1 \leftarrow u_1\\
&\mathit{rain} : 1 \leftarrow \mathit{cloudy}, u_2\\
&\mathit{sprinkler} : 1 \leftarrow \neg\mathit{cloudy}, u_3\\
&\mathit{sprinkler} : 1 \leftarrow \mathit{cloudy}, u_4\\
&\mathit{wet} : 1 \leftarrow \mathit{rain}, u_5\\
&\mathit{wet} : 1 \leftarrow \mathit{sprinkler}, u_6\\
&\mathit{slippery} : 1 \leftarrow \mathit{wet}, u_7
\end{aligned}$$

Next, we introduce the operator (::)/2 to choose clauses in a logic program with a certain probability. The resulting ProbLog clauses then refer to the subset of ProbLog programs **P** in which each clause $C \in \mathrm{LP}(\mathbf{P})$ mentions a unique and distinct error term $u(C)$.

**Example 2.2.48.** Recall the ProbLog program **P** from Example 2.2.43.

**Random facts** Facts(**P**)**:**

$0.5 :: u_1\ 0.6 :: u_2\ 0.7 :: u_3\ 0.1 :: u_4\ 0.8 :: u_5\ 0.3 :: u_6\ 0.9 :: u_7$

**Underlying Logic Program** LP(**P**) :

$$\begin{aligned}
&C_1 := \mathit{cloudy} \leftarrow u_1\\
&C_2 := \mathit{rain} \leftarrow \mathit{cloudy}, u_2\\
&C_3 := \mathit{sprinkler} \leftarrow \neg\mathit{cloudy}, u_3\\
&C_4 := \mathit{sprinkler} \leftarrow \mathit{cloudy}, u_4\\
&C_5 := \mathit{wet} \leftarrow \mathit{rain}, u_5\\
&C_6 := \mathit{wet} \leftarrow \mathit{sprinkler}, u_6\\
&C_7 := \mathit{slippery} \leftarrow \mathit{wet}, u_7
\end{aligned}$$

We observe that each clause $C_i$ in LP(**P**) mentions a unique error term $u_i$. Hence, we independently choose the clause $C_i$ with the probability of random fact $\pi_i :: u_i$. We now denote the ProbLog program **P** with the following ProbLog clauses.

$$\begin{aligned}
&0.5 :: \mathit{cloudy} &&\\
&0.6 :: \mathit{rain} \leftarrow \mathit{cloudy} &&\\
&0.7 :: \mathit{sprinkler} \leftarrow \neg\mathit{cloudy} && 0.1 :: \mathit{sprinkler} \leftarrow \mathit{cloudy}\\
&0.8 :: \mathit{wet} \leftarrow \mathit{rain} && 0.3 :: \mathit{wet} \leftarrow \mathit{sprinkler}\\
&0.9 :: \mathit{slippery} \leftarrow \mathit{wet} &&
\end{aligned}$$

Here, the ProbLog clause $0.6 :: \mathit{rain} \leftarrow \mathit{cloudy}$ for example means that we choose the clause $\mathit{rain} \leftarrow \mathit{cloudy}$ independently with probability 0.6.

Formally, these ProbLog clauses are defined as follows.

**Definition 2.2.33** (ProbLog Clause)**.** A **ProbLog clause** $PC$ is an expression $\pi(PC) :: C(PC)$, where $C(PC)$ is a normal clause and where $\pi(PC) \in [0, 1]$ is a probability. A set of ProbLog clauses **P** denotes the ProbLog program, consisting of the logic program

$$\mathrm{LP}(\mathbf{P}) := \{\mathrm{head}(C(PC)) \leftarrow \mathrm{body}(C(PC)) \cup \{u(PC)\} \mid PC \in \mathbf{P}\}$$

and the random facts

$$\mathrm{Facts}(\mathbf{P}) := \{\pi(PC) :: u(PC) \mid PC \in \mathbf{P}\},$$

where we introduce a distinct error term $u(PC) \notin \mathfrak{P}$ for every clause $PC \in \mathbf{P}$. Each ProbLog program $\mathbf{P}$ that can be represented in this way is said to be contained in the **fragment of ProbLog clauses**. In this case, we define the **dependence graph** graph($\mathbf{P}$) of $\mathbf{P}$ to be the directed graph on the alphabet $\mathfrak{P}$ that is given by drawing an edge $p \rightarrow q$ if and only if there exists a clause $PC \in \mathbf{P}$ with head$(PC) = q$ and with $\{p, \neg p\} \cap \text{body}(PC) \neq \emptyset$. Finally, we say that $\mathbf{P}$ is **acyclic** if the graph graph($\mathbf{P}$) contains no cycle.

The ProbLog clause notation suggests that in a ProbLog program $\mathbf{P}$ within the fragment of ProbLog clauses, the uncertainty can be interpreted as uncertainty about the clauses in the logic program $\mathbf{P}' := \{C(PC)|\ PC \in \mathbf{P}\}$. More precisely, each clause $C(PC) \in \mathbf{P}'$ is chosen independently with a probability of $\pi(PC)$.

Finally, let $\mathfrak{A} := (\mathcal{C}, \mathcal{P})$ be a relational alphabet. In this case, a **random fact** is an expression $\pi(RF) :: u(RF)$, where the **error term** $u(RF) := p(t_1, ..., t_n)$ is an atom with underlying predicate $p \notin \mathcal{P}$ and where $\pi(RF) \in [0, 1]$ is the **probability** of $u(RF)$. For a ground substitution $\gamma$, we define $RF^\gamma := (\pi(RF) :: u(RF)^\gamma)$ to be the corresponding **ground instance**. If $\Phi$ is a set of random facts, we denote by $\bar{\Phi}$ the set of all ground instances $RF^\gamma$ where $RF \in \Phi$.

Now, a ProbLog program $\mathbf{P}$ consists of a finite set of random facts Facts($\mathbf{P}$) and a logic program LP($\mathbf{P}$) in the alphabet $\bar{\tilde{\mathfrak{A}}} := (\mathcal{C}, \mathcal{P} \cup \{p\text{: } p \text{ underlying predicate of } u(RF),\ RF \in \text{Facts}(\mathbf{P})\})$ such that the **grounding** $\bar{\mathbf{P}}$ given by $\overline{\text{Facts}(\mathbf{P})}$ and $\overline{\text{LP}(\mathbf{P})}$ is a ProbLog program in the propositional alphabet $\mathfrak{P}(\mathfrak{A})$ of all ground atoms in $\mathfrak{A}$. In this case, the probability of a Herbrand structure $\omega$ is also defined by grounding, i.e. $\pi(\omega) := \pi_{\mathbf{P}}(\omega) := \pi_{\bar{\mathbf{P}}}(\omega)$.

**Example 2.2.49.** Reconsider the alphabet $\mathfrak{A} := (\mathcal{P}, \mathcal{C})$ in Example 2.2.37 with predicates

$$\mathcal{P} := \{smokes, spontaneously\_smokes\}$$

and constants

$$\mathcal{C} := \{jakob, kilian\}.$$

Each predicate has arity one, and the constants denote two friends, Jakob and Kilian, respectively. Furthermore, *spontaneously_smokes*(*jakob*) and *spontaneously_smokes*(*kilian*) indicate that Jakob and Kilian start to smoke, respectively. By *smokes*(*jakob*) and *smokes*(*kilian*), we mean that Jakob and Kilian are smoking, respectively. Assume everyone starts spontaneously to smoke with a probability of 0.5. This could result in the ProbLog program $\mathbf{P}$ below.

$$\begin{aligned}
&0.5 :: spontanously\_smokes(X) \\
&smokes(X) \leftarrow spontanously\_smokes(X) \\
&smokes(Y) \leftarrow smokes(X)
\end{aligned}$$

In this case, we find that Jakob is smoking with a probability of $\pi(smokes(jakob)) := 3/4$.

Furthermore, a **ProbLog clause** $PC$ is an expression of the form $\pi(PC) :: C(PC)$ for a normal clause $C(PC)$ and a probability $\pi(PC) \in [0, 1]$. For a ground substitution $\gamma$, the corresponding **ground instance** is defined to be $PC^\gamma := \pi(PC) :: C(PC)^\gamma$. Finally, we identify a finite set of ProbLog clauses $\mathbf{P}$ with the set $\bar{\mathbf{P}}$ of all ground instances $PC^\gamma$ of ProbLog clauses $PC \in \mathbf{P}$, which denotes a ProbLog program in the propositional alphabet $\mathfrak{P}(\mathfrak{A})$ of all ground atoms in $\mathfrak{A}$. Again, the probability of a Herbrand structure $\omega$ is defined by grounding, i.e. $\pi(\omega) := \pi_{\mathbf{P}}(\omega) := \pi_{\bar{\mathbf{P}}}(\omega)$.

**Example 2.2.50.** If we are unsure whether Jakob and Kilian influence each other to smoke in the program $\mathbf{P}$ of Example 2.2.49, this could result in the following program.

$$\begin{aligned}
&0.5 :: spontanously\_smokes(X) \\
&1 :: smokes(X) \leftarrow spontanously\_smokes(X) \\
&0.8 :: smokes(Y) \leftarrow smokes(X)
\end{aligned}$$

In this case, we find that Jakob is smoking with a probability of $\pi(smokes(jakob)) := 7/10$.

### The Language LP$^{\text{MLN}}$

Lee and Wang (35) introduce LP$^{\text{MLN}}$ programs to subsume widespread formalisms in statistical relational artificial intelligence. We extend their formalism in Section 3.2 to a general framework tailored to causal reasoning, which also encompasses widespread formalisms in statistical relational artificial intelligence: Markov networks (46), logic programs with annotated disjunctions (62), and ProbLog programs (21; 26).

Inspired by Richardson and Domingos' approach to constructing LogLinear models (46), Lee and Wang (35) extend logic programming by introducing degrees of certainty. This leads to LP$^{\text{MLN}}$ programs, where LP is a shorthand for "logic program" and MLN is a shorthand for "Markov logic network".

**Definition 2.2.34** (Weighted Rule, LP$^{\text{MLN}}$ Program)**.** A **weighted rule** $(w, C)$ is a tuple of a normal clause $C$ and a **weight** $w \in \mathbb{R} \cup \{\infty\}$. An LP$^{\text{MLN}}$ **program P** then is a finite set of weighted rules. In this case, we call $\text{LP}(\mathbf{P}) := \{C : \exists_w (w, C) \in \mathbf{P}\}$ the **underlying logic program**.

Further, let $\omega$ be a structure. We denote by $\mathbf{P}|_\omega := \{(w, C) \in \mathbf{P} : \omega \models C\}$ the program consisting of all the weighted rules that are satisfied in $\omega$. Now, a **stable model** of **P** is a structure $\omega$ that is a stable model of the logic program $\text{LP}(\mathbf{P}|_\omega)$ such that $\omega \models \mathbf{C}$ for all weighted clauses $(\infty, C) \in \mathbf{P}$ with infinite weight, that is, $C \in \mathbf{P}_\omega$ for all weighted clauses $(\infty, C) \in \mathbf{P}$. As in Definition 2.2.10, we associate to every stable model $\omega$ of **P** the weight

$$w(\omega) := w_{\mathbf{P}}(\omega) := \prod_{\substack{(w,C)\in\mathbf{P} \\ w\neq\infty \\ \omega\models C}} \exp(w) = \prod_{\substack{(w,C)\in\mathbf{P}|_\omega \\ w\neq\infty}} \exp(w) = \exp\left(\sum_{\substack{(w,C)\in\mathbf{P}|_\omega \\ w\neq\infty}} w\right).$$

Again, the weight of a formula $\phi$ is given by

$$w(\phi) := w_{\mathbf{P}}(\phi) = \sum_{\substack{\omega \text{ stable model} \\ \omega\models\phi}} w(\omega).$$

Finally, we associate to every stable model or formula _ the probability

$$\pi(\_) := \pi_{\mathbf{P}}(\_) := \frac{w(\_)}{w(\top)}.$$

**Example 2.2.51.** Recall the program $\mathbf{P}_2$ of Example 2.2.27.

$$\begin{aligned}
&\mathit{rain} \leftarrow \mathit{cloudy}\\
&\mathit{sprinkler} \leftarrow \neg\mathit{cloudy}\\
&\mathit{wet} \leftarrow \mathit{rain}\\
&\mathit{wet} \leftarrow \mathit{sprinkler}\\
&\mathit{slippery} \leftarrow \mathit{wet}
\end{aligned}$$

Introducing uncertainty could lead to the following LP$^{\text{MLN}}$ program **P**.

$$\begin{aligned}
&(0, \mathit{cloudy})\\
&(\ln(3), \mathit{rain} \leftarrow \mathit{cloudy})\\
&(\ln(5), \mathit{sprinkler} \leftarrow \neg\mathit{cloudy})\\
&(\infty, \mathit{wet} \leftarrow \mathit{rain})\\
&(\infty, \mathit{wet} \leftarrow \mathit{sprinkler})\\
&(\ln(2), \mathit{slippery} \leftarrow \mathit{wet})
\end{aligned}$$

We find that $\mathbf{P}$ has the stable models

$$\begin{aligned} \omega_1 &:= \emptyset & \omega_2 &:= \{\mathit{cloudy}\} \\ \omega_3 &:= \{\mathit{sprinkler}, \mathit{wet}\} & \omega_4 &:= \{\mathit{sprinkler}, \mathit{wet}, \mathit{slippery}\} \\ \omega_5 &:= \{\mathit{cloudy}, \mathit{rain}, \mathit{wet}\} & \omega_6 &:= \{\mathit{cloudy}, \mathit{rain}, \mathit{wet}, \mathit{slippery}\} \end{aligned}$$

with weights $w(\omega_1) = 6$, $w(\omega_2) = 10$, $w(\omega_3) = w(\omega_5) = 15$ and $w(\omega_4) = w(\omega_6) = 30$. Hence, we find $\pi(\omega_1) = 3/53$, $\pi(\omega_2) = 5/53$, $\pi(\omega_3) = \pi(\omega_5) = 15/106$ and $\pi(\omega_4) = \pi(\omega_6) = 15/53$. In particular, we find the road slippery with a probability of $\pi(\mathit{slippery}) = 30/53$.

**Remark 2.2.14.** Note that there is no intuitive interpretation for the soft weights $w \in \mathbb{R}$ of a clause $C$. In contrast, clauses with weight $w = \infty$ are necessarily satisfied.

In the case of a relational alphabet $\mathfrak{A}$, a **weighted rule** $(w, C)$ consists of a weight $w \in \mathbb{R} \cup \{\infty\}$ and a normal clause $C$. Next, for every ground substitution $\gamma$ we obtain a **ground instance** $(w, C^\gamma)$. An LP$^{\text{MLN}}$ **program P** then is a finite set of weighted rules, and the **grounding $\bar{\mathbf{P}}$** of $\mathbf{P}$ is the set of all ground instances of weighted rules in $\mathbf{P}$. Finally, the semantics of $\mathbf{P}$ is the semantics of the grounding $\bar{\mathbf{P}}$, i.e., $\pi_{\mathbf{P}}(\omega) =: \pi_{\bar{\mathbf{P}}}(\omega)$ for every Herbrand structure $\omega$.

**Example 2.2.52.** Reconsider the situation in Example 2.2.37, where we consider the alphabet $\mathfrak{A} := (\mathcal{P}, \mathcal{C})$ that consists of the predicates

$$\mathcal{P} := \{\mathit{smokes}, \mathit{spontaneously\_smokes}\}$$

and the constants

$$\mathcal{C} := \{\mathit{jakob}, \mathit{kilian}\}.$$

Each predicate has arity one, and the constants denote two friends, Jakob and Kilian, respectively. Furthermore, $\mathit{spontaneously\_smokes}(\mathit{jakob})$ and $\mathit{spontaneously\_smokes}(\mathit{kilian})$ indicate that Jakob and Kilian start to smoke, respectively. By $\mathit{smokes}(\mathit{jakob})$ and $\mathit{smokes}(\mathit{kilian})$, we mean that Jakob and Kilian are smoking, respectively. We then build the following program

$$\begin{aligned} &\mathit{smokes}(X) \leftarrow \mathit{spontaneously\_smokes}(X) \\ &\mathit{smokes}(Y) \leftarrow \mathit{smokes}(X) \\ &\mathit{spontaneously\_smokes}(\mathit{kilian}), \end{aligned}$$

indicating that everyone who smokes spontaneously actually smokes, if one friend is smoking, so does the other friend, and Kilian starts smoking spontaneously. Now assume that we are unsure whether Kilian and Jakob influence each other to smoke. This could result in the LP$^{\text{MLN}}$ program $\mathbf{P}$ below.

$$\begin{aligned} &(\infty, \mathit{smokes}(X) \leftarrow \mathit{spontaneously\_smokes}(X)) \\ &(\ln(2), \mathit{smokes}(Y) \leftarrow \mathit{smokes}(X)) \\ &(\infty, \mathit{spontaneously\_smokes}(\mathit{kilian})) \end{aligned}$$

In this case, we obtain the following grounding $\bar{\mathbf{P}}$.

$$\begin{aligned} &(\infty, \mathit{smokes}(\mathit{kilian}) \leftarrow \mathit{spontaneously\_smokes}(\mathit{kilian})) \\ &(\infty, \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{spontaneously\_smokes}(\mathit{jakob})) \\ &(\ln(2), \mathit{smokes}(\mathit{kilian}) \leftarrow \mathit{smokes}(\mathit{kilian})) \\ &(\ln(2), \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{smokes}(\mathit{kilian})) \\ &(\ln(2), \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{smokes}(\mathit{jakob})) \\ &(\ln(2), \mathit{smokes}(\mathit{kilian}) \leftarrow \mathit{smokes}(\mathit{jakob})) \\ &(\infty, \mathit{spontaneously\_smokes}(\mathit{kilian})) \end{aligned}$$

Hence, we find that **P** has the stable models

$$\begin{aligned}\omega_1 &:= \{\mathit{spontaneously\_smokes}(\mathit{kilian}), \mathit{smokes}(\mathit{kilian}), \mathit{smokes}(\mathit{jakob})\}\\ \omega_2 &:= \{\mathit{spontaneously\_smokes}(\mathit{kilian}), \mathit{smokes}(\mathit{kilian})\}\end{aligned}$$

with the weights $w(\omega_1) := 16$ and $w(\omega_2) := 8$. Finally, we find $\pi(\omega_1) := 2/3$ and $\pi(\omega_2) := 1/3$. In particular, we find Jakob smoking with a probability of $\pi(\mathit{smokes}(\mathit{jakob})) := 2/3$.

### Structure Learning of Propositional Probabilistic Logic Programs

Following the overview of Riguzzi (47, §10), we quickly recall some foundations of structure learning for the case of propositional ProbLog programs in the fragment of ProbLog clauses, i.e., we assume that each ProbLog program is given by a set of ProbLog clauses **P**. Generally, in structure learning, one aims to describe a given dataset with a probabilistic logic program, i.e., one needs to determine the probabilities and the structure of this program.

Given suitable data, all those algorithms search the space of programs defined by a language bias and background knowledge with heuristics relying on statistical tests. Here, the **language bias** defining the clause and therefore the program space is given by mode declarations of the form $\mathit{modeb}(*, q)$ and $\mathit{modeh}(*, p)$ as well as declarations of the form $\mathit{determination}(p, q)$ for propositions $p$ and $q$. Here, $\mathit{modeh}(*, q)$ means that the proposition $q$ can occur in the head of a clause, while $\mathit{modeb}(*, q)$ means that the proposition $q$ can occur in the body of a clause. Finally, $\mathit{determination}(p, q)$ means that $q$ may occur in the body of a clause with the head $p$. A positive ProbLog clause $PC$ lies in the language defined by our bias if we declare $\mathit{modeh}(*, \mathrm{head}(PC))$, $\mathit{modeb}(*, b)$ for all $b \in \mathrm{body}(PC)$, and $\mathit{determination}(\mathrm{head}(PC), b)$ for all $b \in \mathrm{body}(PC)$.

Moreover, we can express **background knowledge** in a logic program defining further propositions in terms of the given data. This is also how one can learn non-positive clauses by adding a clause $\mathit{neg\_p} \leftarrow \neg p$ to the background knowledge for every proposition $p$. We call the pair of a language bias and a background knowledge the **setting** $\mathfrak{S}$ for a structure learning algorithm.

**Example 2.2.53.** Recall the program **P** in Example 2.2.48.

$$\begin{aligned}C_1 &:= 0.5 :: \mathit{cloudy}\\ C_2 &:= 0.6 :: \mathit{rain} \leftarrow \mathit{cloudy}\\ C_3 &:= 0.7 :: \mathit{sprinkler} \leftarrow \neg\mathit{cloudy}\\ C_4 &:= 0.1 :: \mathit{sprinkler} \leftarrow \mathit{cloudy}\\ C_5 &:= 0.8 :: \mathit{wet} \leftarrow \mathit{rain}\\ C_6 &:= 0.3 :: \mathit{wet} \leftarrow \mathit{sprinkler}\\ C_7 &:= 0.9 :: \mathit{slippery} \leftarrow \mathit{wet}\end{aligned}$$

To consider the program **P** in our program search, we specify the following setting $\mathfrak{S}$. First, with $\mathit{modeh}(*, \_)$ and $\mathit{modeb}(*, \_)$, we state the propositions that occur in the head and body of a ProbLog clause, respectively.

$\mathit{modeh}(*, \mathit{claudy})$
$\mathit{modeh}(*, \mathit{rain})$
$\mathit{modeh}(*, \mathit{sprinkler})$
$\mathit{modeh}(*, \mathit{wet})$
$\mathit{modeh}(*, \mathit{slippery})$

$\mathit{modeb}(*, \mathit{claudy})$
$\mathit{modeb}(*, \mathit{rain})$
$\mathit{modeb}(*, \mathit{sprinkler})$
$\mathit{modeb}(*, \mathit{wet})$

$\mathit{modeb}(*, \mathit{neg\_claudy})$

Next, with *determination*$(h, b)$ we indicate the propositions $b$ that may occur in the body of a clause with the head $h$.

$$\begin{aligned}
&determination(rain, cloudy)\\
&determination(sprinkler, cloudy)\\
&determination(sprinkler, neg\_cloudy)\\
&determination(wet, rain)\\
&determination(wet, sprinkler)\\
&determination(slippery, wet)
\end{aligned}$$

So far, we only consider positive clauses. To consider the clause $C_3$, we also need to include the background knowledge $\{neg\_cloudy \leftarrow \neg cloudy\}$.

Note that by the *determination*/2 predicate a language bias essentially provides the dependence graph, i.e. the corresponding cause-effect relationships, of the searched program as prior knowledge to the structure learning algorithm.

## 2.3 Formalizing Causality, Knowledge, and Counterfactuals

Here, we study mathematical formalizations of the concepts of causality and Aristotelian knowledge, which we introduce in Section 2.1. Pearl's theory (41) expresses causal relationships in terms of deterministic functional equations, and Aristotelian knowledge then emerges from solving these equations. In contrast, Bochman (10) aims for a logical theory that captures causal reasoning and Aristotelian knowledge. Specifically, he follows Decartes and expresses causal reasoning in rules (32), which aligns his theory with Pearl's causal models and abductive logic programming under the supported model semantics. Furthermore, CP-logic (60) is a causal semantics for logic programs with annotated disjunctions, producing another probabilistic version of Aristotelian knowledge.

### 2.3.1 Pearl's Functional Causal Models

Pearl (41) suggests modeling causal relationships with deterministic functions. Let us illustrate this idea with the following example.

**Example 2.3.1.** We consider a closed gas container with fixed volume $V$. Further, we denote by $T$ the temperature and be $P$ the pressure in the container. Moreover, assume that the gas container can be heated to a temperature $H$. According to the special gas law, this scenario can be described by the equations

$$T := H \qquad\qquad P := k \cdot \frac{T}{V}, \tag{2.9}$$

where $k$ is a rigid constant, such as $k := 0.25$.

Given a concrete volume, for example, $V := v := 10$, and a concrete temperature, for example, $H := h := 20$, we can explicitly determine the pressure $P := 2 \cdot k = 0.5$. However, in this case we do not only know about the pressure of $P = 0.5$, we also have an explanation in terms of temperature $h = 20$ and volume $v = 10$, i.e. we gain Aristotelian knowledge.

The idea of representing causal relationships with deterministic functions is now formally represented by Pearl's structural causal models (41). Since they are the main object in Pearl's theory of causality (41), we recall their definition.

**Definition 2.3.1** (Structural Causal Model (41, §7.1.1))**.** A **structural causal model**

$$\mathcal{M} := (\mathbf{U}, \mathbf{V}, \mathbf{R}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F}),$$

is a tuple, where

$\mathbf{U}$ is a finite set of **external** variables representing the part of the world outside the model

$\mathbf{V}$ is a finite set of **internal** variables determined by the causal relationships in the model

$\mathbf{R}$ is a function assigning to each **variable** $X \in \mathbf{U} \cup \mathbf{V}$ a set $\mathbf{R}(X)$ of **possible values**

Error is a function assigning to each internal variable $V \in \mathbf{V}$ its **error terms** $\mathrm{Error}(V) \subseteq \mathbf{U}$, i.e. the set of external variables $V$ directly depends on

Pa is a function assigning to each internal variable $V \in \mathbf{V}$ its **parents** $\mathrm{Pa}(V) \subseteq \mathbf{V}$, i.e. the set of internal variables $V$ directly depends on

$\mathbf{F}$ is a function assigning to every internal variable $V \in \mathbf{V}$ a map

$$\mathbf{F}(V) := F_V : \prod_{X \in \mathrm{Pa}(V)} \mathbf{R}(X) \times \prod_{U \in \mathrm{Error}(V)} \mathbf{R}(U) \to \mathbf{R}(V),$$

which itself assigns to each value assignments $\mathrm{pa}(V)$ and $\mathrm{error}(V)$ of the parents $\mathrm{Pa}(V)$ and the error terms $\mathrm{Error}(V)$, respectively, a possible value $F_V(\mathrm{pa}(V), \mathrm{error}(V)) \in \mathbf{R}(V)$.

Here, for a subset of variables $\mathbf{X} \subseteq \mathbf{U} \cup \mathbf{V}$, a **value assignment** is a function $\mathbf{x}$ that chooses a possible value $\mathbf{x}(X) \in \mathbf{R}(X)$ for each variable $X \in \mathbf{X}$. In this context, a **situation** is a value assignment $\mathbf{u}$ for the external variables $\mathbf{U}$. Finally, we identify $\mathcal{M}$ with the system of equations

$$\mathcal{M} := \{V := F_V(Pa(V), \mathrm{Error}(V))\}_{V \in \mathbf{V}}.$$

A **solution** $\mathbf{s}$ of $\mathcal{M}$ then is a value assignment on the variables $\mathbf{U} \cup \mathbf{V}$ such that each equation in $\mathcal{M}$ is satisfied.

**Remark 2.3.1.** One can consider the solutions of a structural causal model as Pearl's proposal for the formalization of Aristotelian knowledge.

**Example 2.3.2.** In Example 2.3.1, we define the set of external variables as $\mathbf{U} := \{V, H\}$, the set of internal variables as $\mathbf{V} := \{T, P\}$ and specify the possible values $\mathbf{R}(X) := \mathbb{R}$ for all variables $X \in \mathbf{V} \cup \mathbf{U}$. Furthermore, we set $\mathrm{Pa}(T) := \emptyset$, $\mathrm{Pa}(P) := \{T\}$, $\mathrm{Error}(T) := \{H\}$ and $\mathrm{Error}(P) := \{V\}$. Finally, we associate with the internal variables $T$ and $P$ the deterministic functions

$$\mathbf{F}(T)(H) := F_T(H) := H \text{ and } \mathbf{F}(P)(T, V) := F_P(T, V) := k \cdot \frac{T}{V}, \text{ respectively.}$$

Overall, $\mathcal{M} := (\mathbf{U}, \mathbf{V}, \mathbf{R}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F})$ yields a structural causal model, which is identified with the system of equations (2.9). A situation is then given by a concrete volume, for example, $V := v := 10$, and a concrete temperature, for example, $H := h := 20$, which allows us to explicitly determine the pressure $P := 0.5$ in Example 2.3.1.

**Remark 2.3.2.** Note that the parents $\mathrm{Pa}(V)$ and the error terms $\mathrm{Error}(V)$ of an internal variable $V \in \mathbf{V}$ can usually be read from the defining function $F_V$. Hence, in the following, we will not explicitly specify the parent map Pa and error term map Error.

In this thesis, we focus mainly on Boolean structural causal models.

**Definition 2.3.2.** A **Boolean** structural causal model $\mathcal{M} := (\mathbf{U}, \mathbf{V}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F})$ is a structural causal model $\mathcal{M} = (\mathbf{U}, \mathbf{V}, \mathbf{R}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F})$ with $\mathbf{R}(X) := \{True, False\}$ for every $X \in \mathbf{U} \cup \mathbf{V}$.

**Example 2.3.3.** Consider a road that passes through a field with a sprinkler in it. The sprinkler is turned on, written *sprinkler*, whenever a weather sensor is activated, written *sensor*. Further, it may rain, denoted by *rain*. If it rains or the sprinkler is on, the pavement of the road gets wet, denoted by *wet*. Finally, a wet road is slippery, denoted by *slippery*.

This mechanism can be represented by a Boolean structural causal model $\mathcal{M}$ with internal variables $\mathbf{V} := \{sprinkler, wet, slippery\}$, with external variables $\mathbf{U} := \{sensor, rain\}$ and with the functions, given by the equations

$$sprinkler := sensor \qquad wet := rain \vee sprinkler \qquad slippery := wet.$$

We are interested in structural causal models because they allow us to represent the effects of external interventions.

### The Effect of External Interventions and Potential Outcomes

Consider a fixed structural causal model $\mathcal{M} := (\mathbf{U}, \mathbf{V}, \mathbf{R}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F})$. Assuming we are given a subset of internal variables $\mathbf{I} \subseteq \mathbf{V}$ along with a value assignment $\mathbf{i}$, we aim to represent the effect of forcing the variables in $\mathbf{I}$ to attain the values specified by $\mathbf{i}$. For this purpose, we construct the **modified (causal) model** or **submodel**

$$\mathcal{M}_{\mathbf{i}} := (\mathbf{U}, \mathbf{V}, \mathbf{R}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F}_{\mathbf{i}}).$$

This is done by replacing the map $\mathbf{F}$ with the map $\mathbf{F}_{\mathbf{i}}$, which is defined by setting

$$\mathbf{F}_{\mathbf{i}}(V)(\mathrm{pa}(V), \mathrm{error}(V)) := \begin{cases} \mathbf{i}(V), & \text{if } V \in \mathbf{I} \\ \mathbf{F}(V)(\mathrm{pa}(V), \mathrm{error}(V)), & \text{else} \end{cases}$$

for every internal variable $V \in \mathbf{V}$ with value assignments $\mathrm{pa}(V)$ and $\mathrm{error}(V)$ for the parents $\mathrm{Pa}(V)$ and the error terms $\mathrm{Error}(V)$, respectively. The idea here is that the modified model $\mathcal{M}_{\mathbf{i}}$ represents a minimal change to the model $\mathcal{M}$ that forces the variables in $\mathbf{I}$ to attain the values specified by $\mathbf{i}$ (41, Ch. 7).

**Example 2.3.4.** If we decide to switch the sprinkler on in the causal model $\mathcal{M}$ of Example 2.3.3

$$sprinkler := sensor \qquad wet := rain \vee sprinkler \qquad slippery := wet,$$

we obtain the modified model $\mathcal{M}_{sprinkler} := \mathcal{M}_{sprinkler:=True}$ that is given by the following equations.

$$sprinkler := True \qquad wet := rain \vee sprinkler \qquad slippery := wet$$

**Notation 2.3.1.** Let $V$ be an internal variable of a Boolean structural causal model $\mathcal{M}$. In this case, we write $\mathcal{M}_V := \mathcal{M}_{V:=True}$ and $\mathcal{M}_{\neg V} := \mathcal{M}_{V:=False}$.

As in Example 2.3.4 our actions often force a variable in a causal model to attain a new value. Introducing the **do-operator** by setting

$$\mathcal{M}^{\mathrm{do}(\mathbf{i})} := \mathcal{M}_{\mathbf{i}}$$

for a structural causal model $\mathcal{M}$ and a value assignment on internal variables $\mathbf{i}$, Pearl (41) emphasizes that submodels $\mathcal{M}^{\mathrm{do}(\mathbf{i})}$ often result from doing something that forces some variables to values according to the assignment $\mathbf{i}$. To obtain well-defined results, Pearl (41) restricts himself to the study of functional causal models.

**Definition 2.3.3** (Functional Causal Model and Potential Response)**.** A **(functional) causal model** $\mathcal{M} := (\mathbf{U}, \mathbf{V}, \mathbf{R}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F})$ is a structural causal model, in which for each value assignment $\mathbf{i}$ to a subset of internal variables $\mathbf{I} \subseteq \mathbf{V}$, every situation $\mathbf{u}$ of $\mathcal{M}$ yields a unique solution $\mathbf{s}_{\mathbf{i}}(\mathbf{u})$ of the modified model $\mathcal{M}_{\mathbf{i}}$.

In this case, we write $Y_{\mathbf{i}}(\mathbf{u})$ for the value that an internal variable $Y \in \mathbf{V}$ attains in the solution $\mathbf{s}_{\mathbf{i}}(\mathbf{u})$ of the modified model $\mathcal{M}_{\mathbf{i}}$ provided that the external variables take the values specified by the situation $\mathbf{u}$. We call $Y_{\mathbf{i}}(\mathbf{u}) \in \mathbf{R}(Y)$ the **potential response** of $Y$ to the intervention of forcing the variables in $\mathbf{I}$ to attain the values specified by $\mathbf{i}$ in the situation $\mathbf{u}$. Finally, $Y(\mathbf{u}) \in \mathbf{R}(Y)$ denotes the value that the internal variable $Y \in \mathbf{V}$ attains in the model $\mathcal{M}$ if the external variables take the values specified by the situation $\mathbf{u}$.

**Remark 2.3.3.** A potential response $y = Y_{\mathbf{i}}(\mathbf{u})$, is interpreted as the counterfactual statement:

"The variable $Y$ would have attained the value $y$ in the situation $\mathbf{u}$, had we forced the variables in $\mathbf{I}$ to attain the values specified by $\mathbf{i}$."

**Example 2.3.5.** Reconsider the causal model $\mathcal{M}$ in Example 2.3.3

$$\mathit{sprinkler} := \mathit{sensor} \qquad \mathit{wet} := \mathit{rain} \vee \mathit{sprinkler} \qquad \mathit{slippery} := \mathit{wet}.$$

Assume we know that the sensor is not triggered and that it does not rain. This means that we are given the situation $\mathbf{u}$ consisting of $\mathit{sensor} = \mathit{False}$ and $\mathit{rain} = \mathit{False}$. Furthermore, by investigating the modified model $\mathcal{M}_{\mathit{sprinkler}}$ in Example 2.3.4 we find that $\mathit{slippery}_{\mathit{sprinkler}}(\mathbf{u}) = \mathit{True}$ while $\mathit{slippery}(\mathbf{u}) := \mathit{False}$. We interpret these results as the following statements:

> "The road would have been slippery if we had switched on the sprinkler, even though the sensor is not triggered and it is not raining."
>
> "The road is not slippery if the sensor is not triggered and if it does not rain."

### Logic Programs and Structural Causal Models

Rückschloß and Weitkämper (49) notice that the Clark completion (25) essentially associates each logic program $\mathbf{P}$ with a causal model comp($\mathbf{P}$). This then allows them to transfer Pearl's notion of interventions to logic programming. In this way, they recover the notion of intervention proposed by Vennekens et al. (60) and Riguzzi et al. (48) in probabilistic logic programming.

Assume $\mathbf{i}$ is a truth value assignment on a subset of propositions $\mathbf{I} \subseteq \mathfrak{P}$. Now, the effect of forcing the propositions in $\mathbf{I}$ to attain the truth values according to $\mathbf{i}$ is represented by the modified program $\mathbf{P}_{\mathbf{i}} := \mathbf{P}^{\mathrm{do}(\mathbf{i})}$, which results from $\mathbf{P}$ by erasing all clauses with heads in $\mathbf{I}$ and by adding all positive literals $p \in \mathbf{i}$.

**Example 2.3.6.** Recall the program $\mathbf{P}_2$ of Example 2.2.27.

$$\begin{array}{ll} \mathit{rain} \leftarrow \mathit{cloudy} & \\ \mathit{sprinkler} \leftarrow \neg \mathit{cloudy} & \\ \mathit{wet} \leftarrow \mathit{rain} & \mathit{wet} \leftarrow \mathit{sprinkler} \\ \mathit{slippery} \leftarrow \mathit{wet} & \end{array}$$

If we intervene in $\mathbf{P}_2$ and manually turn the sprinkler off (or on), we obtain the following program.

$$\begin{array}{ll} \mathit{rain} \leftarrow \mathit{cloudy} & \\ (\mathit{sprinkler}) & \\ \mathit{wet} \leftarrow \mathit{rain} & \mathit{wet} \leftarrow \mathit{sprinkler} \\ \mathit{slippery} \leftarrow \mathit{wet} & \end{array}$$

### Probabilistic Causal Models

Next, we introduce probabilities into a functional causal model $\mathcal{M}$ and specify a probability distribution over the situations of $\mathcal{M}$. In this way, we obtain a probabilistic version of Aristotelian knowledge.

**Definition 2.3.4.** A **probabilistic (Boolean) causal model** $\mathbb{M} := (\mathcal{M}, \pi)$ is given by a (Boolean) functional causal model $\mathcal{M}$ together with a probability distribution $\pi$ on the situations of $\mathcal{M}$.

**Example 2.3.7.** Modify Example 2.3.3. Consider a road that passes through a field with a sprinkler in it. The sprinkler is turned on, written *sprinkler*, by a weather sensor with probability 0.1 if it is cloudy, denoted by *cloudy* and with probability 0.7 if it is not cloudy. In addition, it rains, denoted by *rain*, with probability 0.6 if the weather is cloudy. If it rains or the sprinkler is on, the pavement of the road gets wet, denoted by *wet*, with probability 0.9. And in the case where the pavement is wet, we observe with a probability of 0.8 that the road is slippery, denoted by *slippery*.

This mechanism can be represented by a Boolean functional causal model, $\mathcal{M}$, with internal variables $\mathbf{V} := \{cloudy, rain, sprinkler, wet, slippery\}$, and external variables $\mathbf{U} := \{u_1, ..., u_6\}$. The functions are then given by the following equations.

$$\begin{aligned} &cloudy := u_1 && \\ &rain := cloudy \wedge u_2 && sprinkler := (cloudy \wedge u_3) \vee (\neg cloudy \wedge u_4) \\ &wet := (rain \vee sprinkler) \wedge u_5 && slippery := wet \wedge u_6 \end{aligned}$$

To represent the uncertainties in our story, we specify the probabilities $\pi(u_1) = 0.5$, $\pi(u_2) = 0.6$, $\pi(u_3) = 0.1$, $\pi(u_4) = 0.7$, $\pi(u_5) = 0.9$ and $\pi(u_6) = 0.8$. Furthermore, asserting that $u_1, ..., u_6$ are mutually independent Boolean random variables defines a unique distribution $\pi$ on the situations of $\mathcal{M}$, resulting in the probabilistic Boolean causal model $\mathbb{M} := (\mathcal{M}, \pi)$.

Here, we assume that uncertainty arises from hidden variables that we do not explicitly model. However, the influence of these hidden variables is encapsulated in the external variables $\mathbf{U} := \{u_1, ..., u_6\}$. For example, the variable $u_3$ summarizes the potential causes of why the sprinkler could be on if it is cloudy. These potential causes, such as sensor failure or children playing and manually switching on the sprinkler, are not explicitly modeled in $\mathcal{M}$. Nevertheless, the potential influence of these missing parameters is represented by the external variables $\mathbf{U}$ and the distribution $\pi$. An advantage of this representation is its similarity to ProbLog, which allows programmers to readily execute computations in Boolean causal models.

Let $\mathbb{M} := (\mathcal{M}, \pi)$ be a probabilistic causal model. Since this implies that $\mathcal{M}$ is a functional causal model, every situation $\mathbf{u}$ corresponds to a unique solution $\mathbf{s}(\mathbf{u})$ of the corresponding system of equations. Hence, by defining

$$\pi_{\mathbb{M}}(\omega) := \begin{cases} \pi(\mathbf{u}), & \text{if } \omega = \mathbf{s}(\mathbf{u}) \\ 0, & \text{else} \end{cases}$$

for every value assignment $\omega$ of the variables $\mathbf{U} \cup \mathbf{V}$, the model $\mathbb{M}$ gives rise to a joint probability distribution of the random variables in $\mathbf{U} \cup \mathbf{V}$.

**Example 2.3.8.** In Example 2.3.7, the causal model $\mathbb{M}$ yields a probability distribution $\pi_{\mathbb{M}}$ on the truth value assignments for the variables

$$\mathbf{U} \cup \mathbf{V} := \{cloudy, rain, sprinkler, wet, slippery, u_1, ..., u_6\}.$$

This allows us, for instance, to calculate the probability $\pi_{\mathbb{M}}(rain)$ that it rains.

$$\pi_{\mathbb{M}}(rain) = \pi(u_1) \cdot \pi(u_2) = 0.5 \cdot 0.6 = 0.3$$

Finally, probabilistic causal models do not only support queries about conditional and unconditional probabilities. They also support two further causal query types: queries for the effect of external interventions and counterfactual queries.

### Predicting the Effect of External Interventions in the Probabilistic Case

Fix a probabilistic causal model $\mathbb{M} := (\mathcal{M}, \pi)$ with external variables $\mathbf{U}$ and internal variables $\mathbf{V}$. Assume we are given a subset of internal variables $\mathbf{I} \subseteq \mathbf{V}$ together with a value assignment $\mathbf{i}$. To calculate the effect of forcing the variables in $\mathbf{I}$ to attain the values specified by $\mathbf{i}$, we build the **modified model** or **submodel** $\mathbb{M}_{\mathbf{i}} := (\mathcal{M}_{\mathbf{i}}, \pi)$. From the modified model $\mathbb{M}_{\mathbf{i}}$ we can then calculate the desired post-interventional probabilities. According to Pearl (41, §1.4.3), we denote the resulting probability distribution by $\pi_{\mathbb{M}}(\_|\operatorname{do}(\mathbf{I} := \mathbf{i})) := \pi_{\mathbb{M}}(\_|\operatorname{do}(\mathbf{i}))$. Here, again Pearl's do-operator $\operatorname{do}(\_)$ indicates that we actively intervene to change our model.

**Example 2.3.9.** We recall Example 2.3.7 and ask for the post-interventional probability that the road is slippery after turning off the sprinkler. In this case, we query the modified model $\mathbb{M}_{\neg sprinkler}$ for *slippery* to obtain the probability

$$\pi_{\mathbb{M}}(slippery|\operatorname{do}(\neg sprinkler)) = \pi(u_1) \cdot \pi(u_2) \cdot \pi(u_5) \cdot \pi(u_6) = 0.216$$

for the road to be slippery after switching the sprinkler off.

Finally, we highlight that this result differs from the probability

$$\pi_{\mathbb{M}}(slippery|\neg sprinkler) = \frac{\pi(u_1) \cdot \pi(u_2) \cdot \pi(\neg u_3) \cdot \pi(u_5) \cdot \pi(u_6)}{\pi(u_1) \cdot \pi(\neg u_3) + \pi(\neg u_1) \cdot \pi(\neg u_4)} = 0.432$$

that it is slippery if we observe the sprinkler to be off. In particular, observing the sprinkler being off enhances the probability that it is cloudy, making rain more probable, while manually turning the sprinkler off does not allow such a conclusion. Therefore, in general, intervening in a model yields outcomes that differ from deriving conclusions solely from observations.

Further, probabilistic causal models allow us to generalize potential responses to treat counterfactuals involving internal variables.

### Counterfactual Reasoning or Reasoning with both Observations and Interventions

We do not only want to either observe or intervene in a probabilistic causal model $\mathbb{M} := (\mathcal{M}, \pi)$. We also would like to reason on what would have happened if we had intervened before observing some evidence. Hence, let $\mathbf{E}, \mathbf{I} \subseteq \mathbf{V}$ be two subsets of the internal variables $\mathbf{V}$. Furthermore, let $\mathbf{q}$ be an event in the probability space generated by the probabilistic causal model $\mathbb{M}$. Now suppose we observe evidence that the variables in $\mathbf{E}$ attain values according to the assignment $\mathbf{e}$ and we ask ourselves what is the probability $\pi_{\mathbb{M}}(\mathbf{q}|\mathbf{e}, \operatorname{do}(\mathbf{i}))$ that $\mathbf{q}$ would be true if we had forced the variables in $\mathbf{I}$ to attain values according to the assignment $\mathbf{i}$. Note that $\mathbf{i}$ and $\mathbf{e}$ may contradict each other. According to Pearl (41, §1.4.4), we answer such a **counterfactual query** in three steps:

1.) In the **abduction** step, we adjust the distribution on the situations $\mathbf{u}$ by replacing $\pi(\_)$ with the conditional distribution $\pi(\_|\mathbf{e})$ yielding the model $(\mathcal{M}, \pi(\_|\mathbf{e}))$.

2.) In the **action** step, we intervene in the resulting model according to the assignment $\mathbf{i}$ yielding the modified model $(\mathcal{M}_{\mathbf{i}}, \pi(\_|\mathbf{e}))$.

3.) In the **prediction** step, we finally compute the desired probability $\pi_{\mathbb{M}}(\mathbf{q}|\mathbf{e}, \operatorname{do}(\mathbf{i}))$ by querying the model $(\mathcal{M}_{\mathbf{i}}, \pi(\_|\mathbf{e}))$ in step 2.) for the probability of $\mathbf{q}$.

In the first step, we restrict ourselves to the possible worlds in which we can observe our evidence. Further, by employing the potential outcome framework, we build hypothetical scenarios from our intervention to finally compute the desired probability.

We call the probability $\pi(\mathbf{q}|\mathbf{e}, \mathrm{do}(\mathbf{i}))$ a **counterfactual probability**, as it denotes our degree of belief in a counterfactual query. We use the notation $\pi(\mathbf{q}|\mathbf{e}, \mathrm{do}(\mathbf{i}))$ of Papantonis and Belle (40) to highlight that counterfactual probabilities are computed using both conditional probabilities and Pearl's interventions.

**Example 2.3.10.** Revisiting Example 2.3.7, let us assume that we observe both rain and slippery road. We are interested in determining the probability

$$\pi\left(slippery|slippery, rain, \mathrm{do}(\neg rain)\right),$$

which quantifies the likelihood of the road being slippery under the counterfactual scenario where it had not rained.

To answer this query, in the abduction step, we first update the distributions on the situations according to the evidence $\mathbf{e} := \{rain = True, slippery = True\}$. In particular, we obtain $\pi(u_1|\mathbf{e}) = 1$, $\pi(u_2|\mathbf{e}) = 1$, $\pi(u_3|\mathbf{e}) = 0.1$, $\pi(u_4|\mathbf{e}) = 0.7$, $\pi(u_5|\mathbf{e}) = 1$ and $\pi(u_6|\mathbf{e}) = 1$. Further, in the action step, we intervene in the resulting causal model according to $rain := False$ and get the modified model

$$\begin{aligned} &cloudy := u_1 && \\ &rain := False && sprinkler := (cloudy \wedge u_3) \vee (\neg cloudy \wedge u_4) \\ &wet := (rain \vee sprinkler) \wedge u_5 && slippery := wet \wedge u_6. \end{aligned}$$

Finally, in the prediction step, we compute the desired probability from this modified causal model. By solving the equations we obtain $slippery = u_3$, which yields the result

$$\pi\left(slippery|slippery, rain, \mathrm{do}(\neg rain)\right) = 0.1.$$

To execute the abduction step, it is necessary to calculate and store the conditional distribution $\pi(\_|\mathbf{e})$ on the situations. To avoid storing and computing the distribution $\pi(\_|\mathbf{e})$, we adopt the twin network method proposed by Balke and Pearl (5). Initially, we create two copies, $\mathbf{V}^r$ and $\mathbf{V}^a$, of the internal variables $\mathbf{V}$. One represents the real world where evidence is observed, and the other represents Lewis' alternative worlds (37). In addition, we set $\mathbf{U}^r := \mathbf{U}^a := \mathbf{U}$ for the external variables $\mathbf{U}$. This enables us to establish canonical mappings $\_^{a/r}$ for subsets of variables $\mathbf{X} \subseteq \mathbf{U} \cup \mathbf{V}$ and their value assignments, respectively.

Next, we construct a new causal model $\mathcal{M}^T$ with internal variables $\mathbf{V}^a \cup \mathbf{V}^r$, external variables $\mathbf{U}$, parents $\mathrm{Pa}(\_)^{a/r}$, and error terms $\mathrm{Error}(\_)^{a/r}$ by defining for every internal variable $V \in \mathbf{V}$:

$$V^a := f_V(\mathrm{pa}(V)^a, \mathrm{error}(V)^a) \qquad V^r := f_V(\mathrm{pa}(V)^r, \mathrm{error}(V)^r)$$

This leads to the **twin model** $\mathbb{M}^T := (\mathcal{M}^T, \pi)$. Following Pearl (41), to construct Lewis' alternative worlds (37) we intervene and force the variables in $\mathbf{I}^a$ to attain the values specified by $\mathbf{i}^a$, yielding the **modified twin model** $\mathbb{M}^T_{\mathbf{i}^a} := (\mathcal{M}^T_{\mathbf{i}^a}, \pi)$. One then expects that

$$\pi_{\mathbb{M}}(\mathbf{q}|\mathbf{e}, \mathrm{do}(\mathbf{i})) = \pi_{\mathbb{M}^T_{\mathbf{i}^a}}(\mathbf{q}^a|\mathbf{e}^r).$$

**Remark 2.3.4.** Balke and Pearl (5) informally argue for the correctness of the above equation, a formal proof can be found in the appendix of Kiesel et al. (34).

**Example 2.3.11.** If we again calculate the probability $\pi_{\mathbb{M}}\left(slippery|slippery, rain, \mathrm{do}(\neg rain)\right)$ in Example 2.3.10 using the twin network method, we build the modified twin model $\mathcal{M}^T_{\neg rain^a}$ below.

$$\begin{aligned} &cloudy^r := u_1 && cloudy^a := u_1 \\ &rain^r := cloudy^r \wedge u_2 && rain^a := False \\ &sprinkler^r := (cloudy^r \wedge u_3) \vee (\neg cloudy^r \wedge u_4) && sprinkler^a := (cloudy^a \wedge u_3) \vee (\neg cloudy^a \wedge u_4) \\ &wet^r := (rain^r \vee sprinkler^r) \wedge u_5 && wet^a := (rain^a \vee sprinkler^a) \wedge u_5 \\ &slippery^r := wet^r \wedge u_6 && slippery^a := wet^a \wedge u_6 \end{aligned} \tag{2.10}$$

As desired, we then obtain

$$\pi_{\mathbb{M}}(\mathit{slippery}|\mathit{slippery}, \mathit{rain}, \mathrm{do}(\neg \mathit{rain})) = \pi_{\mathbb{M}^T_{\neg \mathit{rain}^a}}(\mathit{splippery}^a|\mathit{splippery}^r, \mathit{rain}^r) = 0.1.$$

### 2.3.2 Causal Queries for Logic Programs with Annotated Disjunctions

In Section 3.4.4, we embed logic programs with annotated disjunctions (62) into our formalism tailored to causal reasoning. This then allows us to transfer our causal reasoning to logic programs with annotated disjunctions.

Vennekens et al. (60) independently introduce CP-logic as a causal semantics for logic programs with annotated disjunctions. The target object of this semantics is represented by processes, which are a generalization of Shafer's event trees (56, Ch. 2). In this thesis, our objective is to align our causal reasoning with their results. Therefore, we do not delve into the specific construction of this semantics. Instead, we focus on its realization concerning the causal query types, that is, predicting the effect of external intervention and counterfactual queries, as outlined by Vennekens et al. (61) and Riguzzi et al. (48).

Let $\mathbf{P}$ be an LPAD in a propositional alphabet $\mathfrak{P}$, and let $\mathbf{i}$ be a truth value assignment on a subset of propositions $\mathbf{I} \subseteq \mathfrak{P}$. In order to represent the intervention of forcing the propositions in $\mathbf{I}$ to attain the truth values specified by the assignment $\mathbf{i}$, Vennekens et al. (61) and Riguzzi et al. (48) build the **modified program** $\mathbf{P_i} := \mathbf{P}^{\mathrm{do}(\mathbf{i})}$: The program $\mathbf{P_i}$ results from $\mathbf{P}$ by erasing all head atoms $h_i \in \mathbf{I}$ from the clauses $C \in \mathbf{P}$ and adding a fact $p : 1 \leftarrow$ for every positive literal $p \in \mathbf{i}$. Finally, for a formula $\phi$ the probability

$$\pi_{\mathbf{P_i}}(\phi) := \pi_{\mathbf{P}}(\phi| \, \mathrm{do}(\mathbf{i}))$$

is interpreted as the probability of $\phi$ to hold after intervening according to $\mathbf{i}$.

**Example 2.3.12.** If we switch off the sprinkler in the LPAD of Example 2.2.41

$$\begin{aligned}
&\mathit{cloudy} : 0.5\\
&\mathit{rain} : 0.6; \mathit{sprinkler} : 0.1 \leftarrow \mathit{cloudy}\\
&\mathit{sprinkler} : 0.8 \leftarrow \neg \mathit{cloudy}\\
&\mathit{wet} : 0.7 \leftarrow \mathit{rain}\\
&\mathit{wet} : 0.3 \leftarrow \mathit{sprinkler}\\
&\mathit{slippery} : 0.9 \leftarrow \mathit{wet},
\end{aligned}$$

we obtain the following modified program.

$$\begin{aligned}
&\mathit{cloudy} : 0.5\\
&\mathit{rain} : 0.6 \leftarrow \mathit{cloudy}\\
&\leftarrow \neg \mathit{cloudy} && \text{(No selection chooses this clause)}\\
&\mathit{wet} : 0.7 \leftarrow \mathit{rain}\\
&\mathit{wet} : 0.3 \leftarrow \mathit{sprinkler}\\
&\mathit{slippery} : 0.9 \leftarrow \mathit{wet}
\end{aligned}$$

In this case, we find that the road is slippery with a probability of

$$\pi(\mathit{slippery}| \, \mathrm{do}(\neg \mathit{sprinkler})) = 0.5 \cdot 0.6 \cdot 0.7 \cdot 0.9 = 0.189.$$

Next, assume that we additionally observe the evidence that the propositions $\mathbf{E} \subseteq \mathfrak{P}$ attain truth values according to the assignment $\mathbf{e}$. We are interested in the counterfactual probability $\pi(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$ that a formula $\phi$ would hold if we had intervened according to $\mathbf{i}$ before observing the evidence $\mathbf{e}$. To answer this counterfactual query, Vennekens et al. (61) use the following procedure.

**Procedure 2.3.1** (Treatment of Counterfactuals in CP-logic)**.** Suppose that the logic program $\mathbf{P}^\sigma$ has a unique stable model $\omega(\sigma)$ for every selection $\sigma$ of $\mathbf{P}$. We intervene in the logic program $\mathbf{P}^\sigma$ according to $\mathbf{i}$ and obtain the logic program $\mathbf{P}^\sigma_\mathbf{i}$. Assuming that the program $\mathbf{P}^\sigma_\mathbf{i}$ has a unique stable model $\omega_\mathbf{i}(\sigma)$ for every selection $\sigma$, we define

$$\pi^\sigma(\phi) = \begin{cases} 1, & \omega(\sigma) \models \mathbf{e} \text{ and } \omega_\mathbf{i}(\sigma) \models \phi \\ 0, & \text{else} \end{cases}.$$

Finally, we express

$$\pi_\mathbf{P}(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i})) := \sum_{\sigma \text{ selection of } \mathbf{P}} \pi^\sigma(\phi) \cdot \frac{\pi(\sigma)}{\pi_\mathbf{P}(\mathbf{e})}. \tag{2.11}$$

**Example 2.3.13.** Recall the LPAD $\mathbf{P}$ in Example 2.2.41

$$\begin{aligned} &\mathit{cloudy} : 0.5 \\ &\mathit{rain} : 0.6; \mathit{sprinkler} : 0.1 \leftarrow \mathit{cloudy} \\ &\mathit{sprinkler} : 0.8 \leftarrow \neg \mathit{cloudy} \\ &\mathit{wet} : 0.7 \leftarrow \mathit{rain} \\ &\mathit{wet} : 0.3 \leftarrow \mathit{sprinkler} \\ &\mathit{slippery} : 0.9 \leftarrow \mathit{wet} \end{aligned}$$

and consider the probability

$$\pi(\mathit{slippery}|\mathit{sprinkler}, \mathit{cloudy}, \mathrm{do}(\neg \mathit{sprinkler})),$$

which represents the likelihood of the road being slippery if we had switched the sprinkler off before observing that the sprinkler is indeed on while it is cloudy. In this case, we examine the selections $\sigma$ that choose the following clauses for $\mathbf{P}^\sigma$.

$$\begin{aligned} &\mathit{cloudy} \\ &\mathit{sprinkler} \leftarrow \mathit{cloudy} \\ &\mathit{wet} \leftarrow \mathit{sprinkler} \\ &\mathit{slippery} \leftarrow \mathit{wet} \end{aligned}$$

Specifically, we find that it is not raining, since we never select the clause $\mathit{rain} \leftarrow \mathit{cloudy}$. Therefore, manually turning off the sprinkler certainly results in a dry road that is not slippery. Consequently, we find

$$\pi(\mathit{slippery}|\mathit{sprinkler}, \mathit{cloudy}, \mathrm{do}(\neg \mathit{sprinkler})) = 0.$$

### 2.3.3 Bochman's Logical Theory of Causality

Bochman's logical theory of causality (10) sets the formal frame of the theory presented in this thesis. In this section, we provide an overview of this logical framework designed for the study of causality and Aristotelian knowledge.

#### A Motivation of Bochman's Framework

Bochman (10) proposes a formalization of causal reasoning and Aristotelian knowledge as introduced in Chapter 2.1. He considers an agent who employs causal reasoning to make sense of **factual knowledge** about the world, that is, knowledge about what is. Such factual knowledge can, for instance, be observed or logically derived from other factual knowledge. Causal reasoning then becomes a binary relation that explains factual knowledge through other factual knowledge. The resulting causal explanations are then either infinite regressions or they begin with factual knowledge that is judged to be self-explained. Finally, **Aristotelian knowledge** is the subset of factual knowledge that can be explained by causal reasoning.

**Example 2.3.14.** We may observe that a lightning strike hits a house and that this house burns down, that is, we have factual knowledge about the lightning strike hitting the house and the house burning down. If we decide further that the information on the lightning strike does not need further explanation, i.e. we accept it as self-explained a priori knowledge, we can use the lightning strike to explain the observation of the house burning down. This leads us to the following conclusion: "Lightning strikes cause houses to burn down."

Everyday causal reasoning also behaves well with logical inference in the causes of events.

**Example 2.3.15.** If we accept the statement "Smoking or genetic predispositions may cause cancer", we typically also accept the statement "Smoking may cause cancer" because "Smoking or genetic predispositions" is a consequence of "Smoking".

Furthermore, causal reasoning behaves well with logical inference in the effect of events.

**Example 2.3.16.** In the scenario of Example 2.3.14, it is reasonable to assume the self-explained factual knowledge that a house burning down usually includes the knowledge about a fire in the bedroom of this house. As a fire in the bedroom is a logical consequence of the house burning down in Example 2.3.14, we would deduce the statement "Lightning strikes cause fire in bedrooms".

Bochman (10) now formalizes observations and factual knowledge with propositional formulas in an alphabet $\mathfrak{P}$.

**Example 2.3.17.** To describe Examples 2.3.14 and 2.3.16 we consider the alphabet

$$\mathfrak{P} := \{lightning, arson, fire, fire\_bedroom\}.$$

Here, $lightning$ denotes the event that a lightning strike hits a house, $arson$ denotes the event that arson takes place, $fire$ denotes the event of a house burning down and $fire\_bedroom$ denotes the event of fire in the bedroom of this house.

We may, for example, assume the factual background knowledge $fire \rightarrow lighting \vee arson$. This means we have the factual knowledge that a house burning down usually includes knowledge about a lightning strike hitting the house or arson. If we now observe that a house is burning down, that is, if we observe $fire$, we conclude that a lightning hit the house or that arson took place, that is, $arson \vee fire$. However, we would not accept $fire$ as an explanation for these events; that is, we have factual knowledge about arson or lightning strike but not Aristotelian knowledge.

Next, Bochman (10) formalizes causal reasoning as a binary relation $(\Rightarrow)/2$ on factual knowledge, that is, on propositional formulas.

**Example 2.3.18.** In Example 2.3.17, causal reasoning is given by a binary relation $(\Rightarrow)/2$ on the formulas in the alphabet $\mathfrak{P} := \{lightning, arson, fire, fire\_bedroom\}$. Specifically, for two propositional formulas $\phi$ and $\psi$ we find $\phi \Rightarrow \psi$ whenever we would accept $\phi$ being true as a causal explanation for $\psi$ being true, that is, if we would accept $\phi$ as a cause of $\psi$. Hence, among other relationships we may find:

$arson \Rightarrow fire$ (arson causes houses to burn down)

$arson \Rightarrow fire\_bedroom$ (arson causes fire in bedrooms)

$arson \Rightarrow fire \wedge fire\_bedroom$ (arson causes houses burning down and fire in bedrooms)

$lightning \vee arson \Rightarrow fire$ (lightnings or arson cause houses burning down)

$lightning \Rightarrow lightning$ (lightnings are self-explained)

$lightning \vee arson \Rightarrow fire \wedge fire\_bedroom$
(lightnings or arson cause houses burning down and fire in bedrooms)

$(fire \rightarrow fire\_bedroom) \Rightarrow (fire \rightarrow fire\_bedroom)$
(concluding from a house burning down to a fire in the bedroom of this house is self-explained)

We highlight that causal reasoning does not lead to a new logical connective; that is, nested expressions like $arson \Rightarrow fire \Rightarrow fire\_bedroom$ have no meaning.

Examples 2.3.15 and 2.3.16 illustrate that causal reasoning $(\Rightarrow)/2$ satisfies the following law.

**Law 5.** *Causal reasoning* $(\Rightarrow)/2$ *respects logical inference in causes and effects.*

Law 5 yields that causal reasoning $(\Rightarrow)/2$ is represented by production inference relations.

**Definition 2.3.5** (Production Inference Relation)**.** A **production inference relation** is a binary relation $(\Rightarrow)/2$ on the set of formulas in the alphabet $\mathfrak{P}$ that satisfies the following assertions for all propositional formulas $\phi$, $\psi$ and $\rho$.

i) If we have $\phi \models \psi$ and $\psi \Rightarrow \rho$, then $\phi \Rightarrow \rho$ follows. **(Strengthening)**

ii) If we have $\phi \Rightarrow \psi$ and $\psi \models \rho$, then $\phi \Rightarrow \rho$ follows. **(Weakening)**

iii) If we have $\phi \Rightarrow \psi$ and $\phi \Rightarrow \rho$, then $\phi \Rightarrow \psi \wedge \rho$ follows. **(And)**

iv) We have $\top \Rightarrow \top$ and $\bot \Rightarrow \bot$. **(Truth and Falsity)**

Note that the propositional formulas $\phi$, $\psi$ and $\rho$ do not mention the binary relation $(\Rightarrow)/2$.

If we find $\phi \Rightarrow \psi$ for two formulas $\phi$ and $\psi$, we say that $\phi$ **causes** $\psi$ or that $\phi$ is a **cause** of $\psi$ or that $\psi$ is an **effect** of $\phi$. Given a production inference relation $(\Rightarrow)/2$, we write $\Phi \Rightarrow \psi$ for a set of propositional formulas $\Phi$ and a formula $\psi$ if there exists a finite subset $\Phi' \subseteq \Phi$ such that $\left(\bigwedge_{\phi \in \Phi'} \phi\right) \Rightarrow \psi$. Furthermore, we define the **consequence operator** $\mathcal{C}$ by assigning to a set of propositional formulas $\Phi$ the set of propositional formulas

$$\mathcal{C}(\Phi) := \{\psi \text{ propositional formula: } \Phi \Rightarrow \psi\}.$$

Again, $\Phi$ and $\mathcal{C}(\Phi)$ are sets of propositional formulas that do not mention the relation $(\Rightarrow)/2$.

The causal operator $\mathcal{C}(\Phi)$ represents the Aristotelian knowledge obtained from causal reasoning $(\Rightarrow)/2$ and factual knowledge about the propositional formulas $\phi \in \Phi$; in other words, $\mathcal{C}(\Phi)$ represents the propositional formulas that are causally explainable with $\Phi$.

**Example 2.3.19.** Recall the alphabet $\mathfrak{P} := \{lightning, arson, fire, fire\_bedroom\}$ of Example 2.3.17. Assume we observe a lightning strike hitting the house that burns down, that is, we observe the event $\Phi := \{lightning, fire\}$. Suppose that the lightning strike is self-explained and causally explains why the house is burning down, that is

$$lightning \Rightarrow lightning \qquad \text{and} \qquad lightning \Rightarrow fire.$$

Furthermore, suppose that the event of the house burning down causally explains a fire in the bedroom of this house, i.e.,

$$fire \Rightarrow fire\_bedroom.$$

Finally, suppose that our causal reasoning cannot decide whether arson took place.

We conclude that

$$lightning, fire, fire\_bedroom \in \mathcal{C}(\Phi) \qquad \text{and} \qquad arson \notin \mathcal{C}(\Phi), \neg arson \notin \mathcal{C}(\Phi).$$

In summary, from Definition 2.3.8 ii) and iii), we conclude that $\mathcal{C}(\Phi)$ is given by the deductive closure of the set of literals $\{lightning, fire, fire\_bedroom\}$, that is,

$$\mathcal{C}(\Phi) = \overline{\{lightning, fire, fire\_bedroom\}}.$$

In the current setting, Bochman (10) further restricts himself to explaining only factual knowledge, that is, causal reasoning only explains events that actually occur.

**Law 6.** *When provided with factual knowledge represented by a deductively closed set of propositional formulas $\Phi$, the Aristotelian knowledge explained by our causal reasoning* $(\Rightarrow)/2$ *constitutes a subset of this factual knowledge, i.e.* $\mathcal{C}(\Phi) \subseteq \Phi$.

This principle leads to the following property of production inference relations.

**Definition 2.3.6** (Regular Production Inference Relation)**.** A **regular** production inference relation $(\Rightarrow)/2$ is a production inference relation that satisfies the following property:

Whenever $\phi \Rightarrow \psi$ and $(\phi \wedge \psi) \Rightarrow \rho$ holds, we find that $\phi \Rightarrow \rho$ also holds. **(Cut)**

**Remark 2.3.5.** Regularity also means that we can substitute other demonstrations in our current demonstration.

As mentioned earlier, regularity formalizes that we only explain factual knowledge.

**Theorem 2.3.1** (Bochman (9, Theorem 8.9))**.** *If and only if a production inference relation is regular, we find* $\mathcal{C}(\Phi) \subseteq \Phi$ *for every deductively closed set of formulas* $\Phi$*. In other words, the production inference relation* $(\Rightarrow)/2$ *satisfies Law 6.* □

Furthermore, the causal reasoning $(\Rightarrow)/2$ adheres to the common laws introduced in Chapter 2.1. Firstly, the causal reasoning $(\Rightarrow)/2$ satisfies Aquinas' *natural necessity*, as stated in Law 1.

**Example 2.3.20.** In Example 2.3.14, this means, for example, that a house will catch fire whenever it is struck by lightning.

Now, suppose that we have $\phi \Rightarrow \rho$ and $\psi \Rightarrow \rho$ for the propositional formulas $\phi$, $\psi$, and $\rho$. Additionally, let us assume that we possess the factual knowledge $\phi \vee \psi$. Hence, we now know that either $\phi$ or $\psi$ is valid, and in both cases, we can explain $\rho$ using Aquinas' *natural necessity* as stated in Law 1.

**Example 2.3.21.** Assuming that both lightning and arson cause a house to burn down, that is, in the language of Example 2.3.17, we obtain the statements:

$$lightning \Rightarrow fire \qquad\qquad arson \Rightarrow fire$$

If we now observe a fire in a house and know that either arson took place or a lightning struck the house, by *natural necessity* in Law 1, we would like to use the disjunction $lightning \vee arson$ to explain $fire$, i.e. we would like to infer

$$lightning \vee arson \Rightarrow fire.$$

We conclude that causal reasoning should be a basic production inference relation.

**Definition 2.3.7** (Basic Production Inference Relation)**.** A **basic** production inference relation $(\Rightarrow)/2$ is one that satisfies the following property.

Whenever $\phi \Rightarrow \rho$ and $\psi \Rightarrow \rho$ we find that $(\phi \vee \psi) \Rightarrow \rho$ is valid. **(Or)**

Now, we have characterized the binary relations on propositional formulas that constitute causal reasoning in the sense of Bochman (10), that is, we are able to define causal production inference relations.

**Definition 2.3.8** (Causal Production Inference Relation)**.** A **causal** production inference relation is one that is both regular and basic.

### Aristotelian Knowledge – The Semantics of Causal Production Inference Relations

Next, we endow (causal) production inference relations with a non-monotonic semantics that represents the possible states of Aristotelian knowledge. In doing so, we rely on Leibniz's *principle of sufficient causation*, as stated in Law 2.

**Example 2.3.22.** In Example 2.3.14, this means, for example, that a fire does not happen without a reason. Therefore, if we observe a fire, it is caused by either a lightning strike or arson.

Now, Theorem 2.3.1 and Leibniz's *principle of sufficient causation* in Law 2 immediately yield the following semantics for (causal) production inference relations.

**Definition 2.3.9** (Exact Theories)**.** An **exact theory** of a (causal) production inference relation $(\Rightarrow)/2$ is a deductively closed set of propositional formulas $\Phi$ such that $\mathcal{C}(\Phi) = \Phi$.

Exact theories represent the potential states of Aristotelian knowledge corresponding to the causal reasoning $(\Rightarrow)/2$. Further, we assume causal reasoning to be complete, i.e., that it explains the truth or falsity of every possible observation.

**Law 7** (Completeness)**.** *Recall from Definition 2.2.9 that a world is a consistent, deductively closed set of propositional formulas $\omega$ that is maximal with respect to inclusion. Every world $\omega$ is either an exact theory or falsified, i.e. $\omega \Rightarrow \bot$.*

In this case, we shall restrict our interest to exact theories that represent complete descriptions of the state of affairs.

**Definition 2.3.10** (Causal Worlds Semantics)**.** A **causal world** of a (causal) production inference relation $(\Rightarrow)/2$ is a world $\omega$ that is an exact theory. We call the set of all causal worlds $\mathrm{Causal}(\Rightarrow)$ the **causal worlds semantics** of $(\Rightarrow)/2$.

**Example 2.3.23.** In the Examples 2.3.14 and 2.3.16, the causal worlds correspond to the following sets of literals.

$$\{\neg lightning, \neg arson, \neg fire, \neg fire\_bedroom\} \qquad \{lightning, \neg arson, fire, fire\_bedroom\}$$
$$\{\neg lightning, arson, fire, fire\_bedroom\} \qquad \{lightning, arson, fire, fire\_bedroom\}$$

### Representing Causal Knowledge through Rules

So far, as illustrated in Example 2.3.18, we represent causal reasoning by stating the whole binary relation of cause and effect. Next, Bochman (10) follows Descartes and expresses causal reasoning through a set of causal rules (32, Ch. 1).

**Definition 2.3.11** (Causal Rules and Causal Theories)**.** A **causal rule** $R$ is an expression of the form

$$\phi \Rightarrow \psi$$

for two propositional formulas $\phi$ and $\psi$, where we call $\phi$ the **cause** and $\psi$ the **effect** of $R$. In addition, a **causal theory** $\Delta$ is a set of causal rules. We denote by $(\Rightarrow_\Delta)/2$ the smallest causal production inference relation extending $\Delta$ and by $\mathcal{C}_\Delta$ the corresponding consequence operator. Observe that $(\Rightarrow_\Delta)/2$ consists of all causal rules following from $\Delta$ with the rules (Strengthening), (Weakening), (And), (Truth and Falsity), (Cut) and (Or) in Definitions 2.3.5, 2.3.6, and 2.3.7, i.e., all rules that apply for the implication in propositional calculus except reflexivity $\phi \to \phi$. A **causal world** of $\Delta$ is a causal world of the production inference relation $(\Rightarrow_\Delta)/2$. Finally, we write $\mathrm{Causal}(\Delta) := \mathrm{Causal}(\Rightarrow_\Delta)$ for the **causal worlds semantics** of $\Delta$.

**Remark 2.3.6.** For any set of propositional formulas $\Phi$, the set $\mathcal{C}_\Delta(\Phi)$ consists of all formulas $\psi$ such that $\Phi \Rightarrow \psi$ can be derived from $\Delta$ using the rules of (Strengthening), (Weakening), (And), (Truth and Falsity), (Cut), and (Or).

**Example 2.3.24.** In the formalism of Example 2.3.17, we consider the following causal theory $\Delta$.

$$\begin{array}{ll}
arson \Rightarrow arson & \neg arson \Rightarrow \neg arson \\
lightning \Rightarrow lightning & \neg lightning \Rightarrow \neg lightning \\
(fire \rightarrow fire\_bedroom) \Rightarrow (fire \rightarrow fire\_bedroom) & \neg fire\_bedroom \Rightarrow \neg fire\_bedroom \\
lightning \Rightarrow fire & \neg fire \Rightarrow \neg fire \\
arson \Rightarrow fire &
\end{array}$$

In this case, the theory $\Delta$ has the causal worlds of Example 2.3.23.

**Example 2.3.25.** Consider the following causal theory $\Delta$.

$$\begin{array}{ll}
(lightning \vee arson) \Rightarrow (lightning \vee arson) & \\
(fire \rightarrow fire\_bedroom) \Rightarrow (fire \rightarrow fire\_bedroom) & \\
lightning \Rightarrow fire & \neg fire \Rightarrow \neg fire \\
arson \Rightarrow fire &
\end{array}$$

Note that $\Delta$ does not make a statement about the proposition *arson*. Hence, the event *arson* cannot be explained by $\Delta$ and therefore *arson* should be false in any causal world of $\Delta$. As the same argument holds also for $\neg arson$, we conclude that there is no causal world of $\Delta$. We conclude further that the causal world semantics is only suitable if we have enough causal knowledge to pin down whole worlds exactly.

As any world is the deductive closure of its literals and as disjunctions in causes of a rule can be split up into different causal rules due to (Or) in Definition 2.3.7, we may restrict ourselves to the study of determinate causal theories.

**Definition 2.3.12** (Literal, Atomic and Determinate Causal Theory)**.** A **literal** causal rule is a causal rule of the form $b_1 \wedge ... \wedge b_n \Rightarrow l$ for literals $b_1, ..., b_n, l$. If, in addition, $l \in \mathfrak{P}$ is an atom, we call the rule **atomic**. Furthermore, a **default** is a rule of the form $\phi \Rightarrow \phi$ for a formula $\phi$ and a **constraint** is a causal rule $b_1 \wedge ... \wedge b_n \Rightarrow \bot$ for literals $b_1, ..., b_n$.

Now, a causal theory $\Delta$ is called **literal** or **atomic** if it only mentions literal or atomic causal rules. An **(atomic) determinate** causal theory $\Delta \cup \mathbf{C}$ is the union of an (atomic) literal causal theory $\Delta$ and a set of constraints $\mathbf{C}$. Lastly, a literal $l$ is a **default** of a determinate causal theory $\Delta$ if $l \Rightarrow l \in \Delta$.

**Remark 2.3.7.** Bochman (10) refers to atomic causal rules and theories by positive literal causal rules and theories, respectively. He further uses the term positive determinate causal theory for an atomic determinate causal theory in our sense.

We obtain the following characterization for the causal worlds of a determinate causal theory (10, §4.5.2).

**Definition 2.3.13** (Completion of a Determinate Causal Theory)**.** The **completion** $\mathrm{comp}(\Delta)$ of a determinate causal theory $\Delta$ is the set of all propositional formulas

$$l \leftrightarrow \bigvee_{\phi \Rightarrow l \in \Delta} \phi,$$

where $l$ is a literal or $\bot$.

**Theorem 2.3.2** (Bochman (9, Theorem 8.115))**.** *The causal world semantics* $\mathrm{Causal}(\Delta)$ *of a determinate causal theory* $\Delta$ *coincides with the set of all models of its completion, i.e.*

$$\mathrm{Causal}(\Delta) := \{\omega \textit{ world: } \omega \models \mathrm{comp}(\Delta)\}. \ \square$$

Further, we obtain the following characterization for causal reasoning that corresponds to literal causal theories.

**Theorem 2.3.3** (Bochman (10, Theorem 4.23))**.** *If $\Delta$ is a literal causal theory, for any literal $l$ and any set of literals $\epsilon$, we find $\epsilon \Rightarrow_\Delta l$ if and only if $\epsilon \Rightarrow l$ is derivable from $\Delta$ using the following inference rules for sets of lierals $\epsilon$, $\epsilon'$ and literals $l$, $l'$:*

*If $\epsilon \Rightarrow l$, then $\epsilon \cup \epsilon' \Rightarrow l$.* ***(Literal Monotonicity)***

*If $\epsilon' \Rightarrow l$ and $\epsilon \cup \{l\} \Rightarrow l'$, then $\epsilon \cup \epsilon' \Rightarrow l'$.* ***(Literal Cut)***

*$\{p, \neg p\} \Rightarrow l'$ for all positive literals $p$ and lierals $l'$* ***(Literals Contradiction)*** □

Next, assume that the production inference relation $(\Rightarrow)/2$ satisfies Law 7 and states complete causal knowledge that determines a set of causal worlds. Let $\omega$ be a world such that $\omega \not\Rightarrow p$ for a proposition $p \in \mathfrak{P}$. In this case, we find either that $\omega$ is a causal world, that is, $\omega \Rightarrow \neg p$, or that $\omega$ is not a causal world, that is, $\omega \Rightarrow \bot$. All in all, we find that the causal world semantics of $(\Rightarrow)/2$ can be given through the negative completion of an atomic determinate causal theory.

**Definition 2.3.14** (Negative Completion and Default Negation)**.** The **negative completion** $\Delta^{nc}$ of the atomic determinate causal theory $\Delta$ is given by

$$\Delta^{nc} := \Delta \cup \{\neg p \Rightarrow \neg p : p \in \mathfrak{P}\}.$$

We say that a causal theory has **default negation** if it is the negative completion of all its atomic causal rules and constraints.

To a causal theory $\Delta$ with default negation, we associate the **causal diagram** graph($\Delta$), which is the directed graph on the propositions $\mathfrak{P}$ that is given by drawing an arrow $p \to q$ if and only if there is a non-default rule $\phi \Rightarrow q \in \Delta$ with $\phi$ mentioning $p$ or $\neg p$. Finally, we say that $\Delta$ is an **acyclic** causal theory if it is a causal theory with default negation whose causal diagram graph($\Delta$) is a directed graph without cycles.

**Remark 2.3.8.** Bochman (10) interprets default rules like $\phi \Rightarrow \phi$ as stating that $\phi$ is self-explained, i.e., factual knowledge about $\phi$ is automatically explained and therefore Aristotelian knowledge. Intuitively, default rules state the knowledge a demonstration must start with.

As default rules play a special role in Bochman's theory (10), only non-default rules induce edges $p \to q$ in the causal diagram graph($\Delta$). In particular, defaults such as $p \Rightarrow p$ or $\neg p \Rightarrow \neg p$ do not induce loops in graph($\Delta$).

If we decide to restrict ourselves to causal theories with default negation, we consider negations as self-evident prior knowledge. This captures the modeling assumption that our parameters come with a default state, which can, without loss of generality, be set to false.

**Example 2.3.26.** For instance, a schedule states the departure of trains or flights, and not when nothing is departing, i.e., the default is that no trains or airplanes depart. Analogously, humans are born as nonsmokers, and the event of them starting to smoke requires an explanation, or houses usually do not burn, i.e., only a fire requires an explanation.

In modeling such scenarios, we initially employ atomic causal rules to identify the direct causes of each proposition $p$. For example, we might assert $arson \Rightarrow fire$ and $lightning \Rightarrow fire$ to indicate that arson or lightning can cause house fires. If these atomic causal rules do not explain the proposition $p$, we interpret this as an explanation for the falsity of $p$, i.e., $\neg p$. In Bochman's framework (10), this principle is captured by forming the negative completion, which additionally states a default $\neg p \Rightarrow \neg p$ for all propositions $p$. Specifically, the default $\neg fire \Rightarrow \neg fire$ implies that houses do not burn unless there is an explanation for a house fire.

In a certain sense, default negation also models the dynamic nature of causality, where we explain how values deviate from their defaults. In summary, causal theories with default negation implement the following assumption.

**Law 8** (Default Negation)**.** *Every proposition $p \in \mathfrak{P}$ is false by default, which means that it is considered false if it cannot be explained by causal reasoning. Hence, only positive literals in a causal world require an explanation.*

Further, we recall the relation between atomic causal theories and Pearl's causal models (41) in Section 2.3.1.

**Definition 2.3.15** (Bochman Translation for Causal Models)**.** Let $\mathcal{M} := (\mathbf{U}, \mathbf{V}, \mathbf{R}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F})$ be a Boolean causal model. The **Bochman translation** of $\mathcal{M}$ is given by the causal theory

$$\Delta(\mathcal{M}) := (\{\mathbf{F}(v) \Rightarrow v \,:\, v \in \mathbf{V}\} \cup \{u \Rightarrow u \,:\, u \in \mathbf{U}\})^{nc}$$

in the alphabet $\mathfrak{P} := \mathbf{U} \cup \mathbf{V}$.

**Example 2.3.27.** For the causal model $\mathcal{M}$ in Example 2.3.3 that is given by the external variables $\mathbf{U} := \{sensor, rain\}$ and the equations

$$sprinkler := sensor \qquad wet := rain \vee sprinkler \qquad slippery := wet,$$

we obtain the Bochman translation $\Delta(\mathcal{M})$

$$\begin{array}{ll}
sensor \Rightarrow sensor & \neg sensor \Rightarrow \neg sensor \\
rain \Rightarrow rain & \neg rain \Rightarrow \neg rain \\
sensor \Rightarrow sprinkler & \neg sprinkler \Rightarrow \neg sprinkler \\
rain \vee sprinkler \Rightarrow wet & \neg wet \Rightarrow \neg wet \\
wet \Rightarrow slippery & \neg slippery \Rightarrow \neg slippery.
\end{array}$$

According to Theorem 2.3.2, determinate causal theories generalize the functional causal models of Pearl (41).

**Corollary 2.3.4.** *The Bochman translation yields a bijection that assigns to every Boolean causal model $\mathcal{M}$ a causal theory $\Delta(\mathcal{M})$ that is equivalent to a causal theory with default negation. The theory $\Delta(\mathcal{M})$ has the same causal diagram, i.e.* $\mathrm{graph}(\mathcal{M}) = \mathrm{graph}(\Delta(\mathcal{M}))$*. Furthermore, for every situation $\mathbf{u}$ of $\mathcal{M}$ the solutions extending $\mathbf{u}$ are exactly the causal worlds $\omega$ of $\Delta(\mathcal{M})$ with $\mathbf{u} \subseteq \omega$.* □

As a consequence of Corollary 2.3.4, we can transfer Pearl's notion of an intervention (41) to causal theories with default negation, i.e. we see that causal reasoning and Aristotelian knowledge are sufficient to represent external interventions.

**Definition 2.3.16** (Intervention in Causal Theories)**.** Assume that we are given a causal theory with default negation $\Delta$ and a truth value assignment $\mathbf{i}$ on a subset of propositions $\mathbf{I} \subseteq \mathfrak{P}$ such that $p \Rightarrow p \notin \Delta$ for all $p \in \mathbf{I}$.

We define the **modified causal theory** $\Delta_{\mathbf{i}}$ by first erasing the atomic causal rules $\phi \Rightarrow p$ explaining propositions $p \in \mathbf{I}$ and replace them with trivial explanations of the form $\top \Rightarrow p$ whenever $p \in \mathbf{i}$, i.e., $p^{\mathbf{i}} = True$.

**Remark 2.3.9.** Bochman (10) initially defines an intervention simply by erasing all rules $\phi \Rightarrow (\neg)p$ for a proposition $p \in \mathfrak{P}$ with $(\neg)p \in \mathbf{i}$ and adding the rules $\top \Rightarrow l$ for all literals $l \in \mathbf{i}$. In our definition, we take into account that we do not consider interventions on external variables in causal models. Furthermore, to ensure that our notion of intervention is well-behaved with the Bochman translations in Definitions 2.3.15 and 2.3.17, we only add the trivial explanation $\top \Rightarrow l$ if $l$ cannot be explained by a default rule in $\Delta$.

As desired, the notion of intervention in Definition 2.3.16 is consistent with the Bochman translation in Definition 2.3.15.

**Proposition 2.3.5.** *Let $\mathcal{M} := (\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{R}, \mathrm{Error}, \mathrm{Pa}, \boldsymbol{F})$ be a Boolean causal model and let $\boldsymbol{i}$ be a truth value assignment on a subset of internal propositions $\boldsymbol{I} \subseteq \boldsymbol{V}$. We find that*

$$\Delta(\mathcal{M}_{\boldsymbol{i}}) \setminus \{\bot \Rightarrow p : p \in \mathfrak{P}\} = \Delta(\mathcal{M})_{\boldsymbol{i}}.$$

*In particular, $\Delta(\mathcal{M}_{\boldsymbol{i}})$ and $\Delta(\mathcal{M})_{\boldsymbol{i}}$ give rise to the same causal worlds.*

*Proof.* First, we observe that the causal theories $\Delta(\mathcal{M}_{\mathbf{i}})$ and $\Delta(\mathcal{M})_{\mathbf{i}}$ have the same defaults. Let $R \in \Delta(\mathcal{M}_{\mathbf{i}})$ be a non-default causal rule. We distinguish the following three cases:

i) The rule $R$ is of the form $\phi \Rightarrow v$ for an internal variable $v \notin \mathbf{I}$. In this case, we conclude that $R$ is of the form $\mathbf{F}(v) \Rightarrow v$.

ii) The rule $R$ is of the form $\phi \Rightarrow v$ for an internal variable $v \in \mathbf{I}$ with $v^{\mathbf{i}} := True$. In this case, $R$ is of the form $\top \Rightarrow v$.

iii) The rule $R$ is of the form $\phi \Rightarrow v$ for an internal variable $v \in \mathbf{I}$ with $v^{\mathbf{i}} := False$. In this case, $R$ is of the form $\bot \Rightarrow v$.

Let $R \in \Delta(\mathcal{M})_{\mathbf{i}}$ be a non-default causal rule. We distinguish the same three cases:

i) The rule $R$ is of the form $\phi \Rightarrow v$ for an internal variable $v \notin \mathbf{I}$. In this case, we conclude that $R$ is of the form $\mathbf{F}(v) \Rightarrow v$.

ii) The rule $R$ is of the form $\phi \Rightarrow v$ for an internal variable $v \in \mathbf{I}$ with $v^{\mathbf{i}} := True$. In this case, $R$ is of the form $\top \Rightarrow v$.

ii) If $v \in \mathbf{V}$ with $v^{\mathbf{i}} = False$, by construction of $\Delta(\mathcal{M})_{\mathbf{i}}$ there is no causal rule of the form $\phi \Rightarrow v$ in $\Delta(\mathcal{M})_{\mathbf{i}}$.

Hence, the desired result follows. □

**Example 2.3.28.** If we manually turn the sprinkler off (or on) in Example 2.3.27, we obtain the causal theory $\Delta(\mathcal{M})_{\neg sprinkler}$ $(\Delta(\mathcal{M})_{sprinkler})$ below.

$$\begin{array}{ll}
sensor \Rightarrow sensor & \neg sensor \Rightarrow \neg sensor \\
rain \Rightarrow rain & \neg rain \Rightarrow \neg rain \\
(\top \Rightarrow sprinkler) & \neg sprinkler \Rightarrow \neg sprinkler \\
rain \vee sprinkler \Rightarrow wet & \neg wet \Rightarrow \neg wet \\
wet \Rightarrow slippery & \neg slippery \Rightarrow \neg slippery
\end{array}$$

Further, we observe $\Delta(\mathcal{M}_{sprinkler}) = \Delta(\mathcal{M})_{sprinkler}$ and

$$\Delta(\mathcal{M}_{\neg sprinkler}) \setminus \{\bot \Rightarrow sprinkler\} = \Delta(\mathcal{M})_{\neg sprinkler}.$$

Finally, we also recall the relation between causal theories with default negation and abductive logic programs.

**Definition 2.3.17** (Bochman Translation for Abductive Logic Programs)**.** To an abductive logic program $\mathcal{P} := (\mathbf{P}, \mathfrak{A}, \mathbf{IC})$ we associate the **Bochman translation**

$$\Delta(\mathcal{P}) := (\{\mathrm{body}(C) \Rightarrow \mathrm{head}(C) : C \in \mathbf{P} \cup \mathbf{IC}\} \cup \{u \Rightarrow u : u \in \mathfrak{A}\})^{nc}.$$

**Example 2.3.29.** For the program $\mathcal{P} := (\mathbf{P}, \mathfrak{A}, \mathbf{IC})$ of Example 2.2.38 consisting of the logic program $\mathbf{P}$

$$\begin{array}{ll} \mathit{rain} \leftarrow \mathit{cloudy} & \\ \mathit{sprinkler} \leftarrow \neg \mathit{cloudy} & \\ \mathit{wet} \leftarrow \mathit{rain} & \mathit{wet} \leftarrow \mathit{sprinkler} \\ \mathit{slippery} \leftarrow \mathit{wet}, & \end{array}$$

the abducibles $\mathfrak{A} := \{\mathit{cloudy}\}$ and the integrity constraint $\mathbf{IC} := \{\bot \leftarrow \neg \mathit{rain}\}$ we obtain the following Bochman translation $\Delta(\mathcal{P})$.

$$\begin{array}{lll} \mathit{cloudy} \Rightarrow \mathit{cloudy} & & \neg \mathit{cloudy} \Rightarrow \neg \mathit{cloudy} \\ \mathit{cloudy} \Rightarrow \mathit{rain} & & \neg \mathit{rain} \Rightarrow \neg \mathit{rain} \\ \neg \mathit{cloudy} \Rightarrow \mathit{sprinkler} & & \neg \mathit{sprinkler} \Rightarrow \neg \mathit{sprinkler} \\ \mathit{rain} \Rightarrow \mathit{wet} & \mathit{sprinkler} \Rightarrow \mathit{wet} & \neg \mathit{wet} \Rightarrow \neg \mathit{wet} \\ \mathit{wet} \Rightarrow \mathit{slippery} & & \neg \mathit{slippery} \Rightarrow \neg \mathit{slippery} \\ \neg \mathit{rain} \Rightarrow \bot & & \end{array}$$

According to Theorem 2.3.2, causal theories with default negation also generalize abductive logic programming under the supported model semantics.

**Corollary 2.3.6.** *The Bochman translation of abductive logic programs is a bijection that associates to every abductive logic program a causal theory with default negation. The causal diagram of the Bochman translation $\Delta(\mathcal{P})$ coincides with the dependence graph of the underlying logic program, and the causal worlds of the Bochman translation $\Delta(\mathcal{P})$ are given by the supported models of $\mathcal{P}$.* □

**Remark 2.3.10.** We further observe that the notion of intervention in a logic program of Section 2.3.1 corresponds to the notion of intervention in a causal theory via the Bochman translation.

To conclude, causal production inference relations formalize causal reasoning as given by the Laws 1, 2, 5 – 8. If we further follow Descartes and represent our causal reasoning through rules, we obtain Bochman's causal theories, which align with Pearl's Boolean causal models and abductive logic programming under the supported model semantics.

In Section 3.1, we begin by observing that Bochman's logical theory of causality (10) yields counterintuitive results once cyclic causal relationships are at play. This behavior is explained by him allowing for cyclic and infinite causal explanations. To circumvent these cyclic and infinite explanations, we extend Bochman's causal theories (10) by explicitly stating the prior knowledge an explanation can start with. This leads us to abductive causal theories, which happen to be equivalent to abductive logic programs under the stable model semantics (27).

Further, in Section 3.2, we combine abductive causal theories with LogLinear models (46) and obtain a formalism tailored to causal reasoning under uncertainty. Lastly, in Section 3.4, we embed widespread frameworks of statistical relational artificial intelligence into this formalism. Transferring our causal reasoning to these frameworks, we finally aim to unify causal reasoning across the realm of statistical relational artificial intelligence.

# Chapter 3

# Boolean Causal Reasoning under Uncertainty

**Statement of Contribution**

This chapter builds on an article submitted for publication (52), Kilian Rückschloß contributed the main idea, definitions, and proofs of all results except Lemma 3.1.5 and 3.1.6, while Felix Weitkämper contributed the condition for consistent counterfactual reasoning in Law 10 and the proofs of Lemma 3.1.5 and 3.1.6.

This chapter improves on that submission with:

- additional examples;
- more detailed proofs;
- a treatment of the non-ground case;
- a characterization of ProbLog as a fragment within weighted abductive logic programming;
- a characterization of ProbLog programs that define unique counterfactual probabilities;
- an adaptation of the twin-network method presented by Kiesel et al. (34);
- a discussion of causality in LPADs as presented by Rückschloß and Weitkämper (50).

In the acyclic case, the Corollaries 2.2.4, 2.3.4, and 2.3.6 show that Bochman's causal theories, Pearl's functional causal models, and abductive logic programs are essentially equivalent formalisms (10). In the cyclic case, however, Pearl's causal models may not be well-defined. On the logic programming side, we have to consider different proposals for the semantics of a logic program, i.e., the supported model semantics (25) for acyclic logic programs, the minimal model semantics (59) for positive logic programs, and the stable model semantics (27) for general logic programs. The following section shows that abductive logic programming (22) under the stable model semantics (27) is the appropriate framework to capture everyday Boolean causal reasoning with default negation.

## 3.1 Deterministic Causal Reasoning

Let us begin by taking a closer look at Bochman's theory of causality (10) as presented in Section 2.3, specifically in the cyclic case.

**Example 3.1.1.** Let *jakob* and *kilian* denote two friends, Jakob and Kilian. Furthermore, *spontaneously_smokes*(*jakob*) and *spontaneously_smokes*(*kilian*) indicate that Jakob and Kilian start to smoke spontaneously. By *smokes*(*jakob*) and *smokes*(*kilian*), we mean that Jakob and Kilian are smoking, respectively. Assuming that humans usually do not smoke as well as that Jakob smokes if Kilian smokes and vice versa, we would model this story in the following causal theory $\Delta$ with default negation:

$$
\begin{aligned}
&\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{spontaneously\_smokes}(\mathit{jakob}) && (3.1)\\
&\mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{spontaneously\_smokes}(\mathit{kilian}) && (3.2)\\
&\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob}) && (3.3)\\
&\mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian}) && (3.4)\\
&\mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob}) && (3.5)\\
&\mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{kilian}) && (3.6)\\
&\neg \mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \neg \mathit{spontaneously\_smokes}(\mathit{jakob})\\
&\neg \mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \neg \mathit{spontaneously\_smokes}(\mathit{kilian})\\
&\neg \mathit{smokes}(\mathit{jakob}) \Rightarrow \neg \mathit{smokes}(\mathit{jakob})\\
&\neg \mathit{smokes}(\mathit{kilian}) \Rightarrow \neg \mathit{smokes}(\mathit{kilian})
\end{aligned}
$$

According to Theorem 2.3.2, we would then obtain a causal world

$$\omega := \{\mathit{smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{kilian})\},$$

where Kilian and Jakob smoke even if nobody smokes spontaneously. We obtain this flaw since Bochman's framework allows for cyclic explanations. The explanation for *smokes*(*kilian*) in $\omega$ is, for instance, given by *smokes*(*kilian*) $\Rightarrow$ *smokes*(*jakob*) and *smokes*(*jakob*) $\Rightarrow$ *smokes*(*kilian*), i.e., Kilian smokes because Jakob smokes, who smokes because Kilian does. We do not accept these cyclic arguments in our daily causal reasoning.

Example 3.1.1 illustrates a potential drawback of Bochman's approach (10), where circular "explanations" may arise in some cases, leading to counterintuitive results. To address this issue, we propose asserting the principle of acyclicity, avoiding cyclic arguments by ensuring that all causal explanations start from specified self-explained a priori knowledge.

**Law 9** (Principle of Acyclicity)**.** *Causal explanations start from self-evident a priori knowledge.*

Interestingly, the principle of acyclicity appears to be relevant only in the Boolean case, as demonstrated by the following example of Pearl (41) and Goldberger (29).

**Example 3.1.2.** Let us consider $q$ as the quantity of household demand for a product, such as espresso machines. We use $p$ to denote the price per unit (that is, per espresso machine), $i$ for household income, $w$ for the wage rate of producing one espresso machine, and $u_{1/2}$ to model some external variables not explicitly considered in the model. An increase in price $p$ or income $i$ leads to a decrease or increase, respectively, in the quantity of household demand $q$. In contrast, a higher quantity of household demand $q$ or a higher wage rate $w$ result in an increase in price $p$. Therefore, we model the espresso machine market with the following causal model

$$q := \alpha \cdot p + \beta \cdot i + u_1 \qquad\qquad p := \gamma \cdot q + \delta \cdot w + u_2.$$

Here, $p$ and $q$ are real-valued internal variables, while $i$, $w$, $u_1$ and $u_2$ are real-valued external variables, and $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ are rigid constants. To determine the resulting price $p'$ and the quantity of household demand $q'$, we solve the system of equations to obtain the following.

$$q' = \frac{\alpha\delta w + \alpha u_2 + \beta i + u_1}{1 - \alpha\gamma} \qquad\qquad p' = \frac{\gamma\beta i + \gamma u_1 + \delta w + u_2}{1 - \alpha\gamma}$$

In particular, neither the resulting price $p'$ nor the quantity of household demand $q'$ require additional justification in terms of external self-evident a priori knowledge.

From the principle of acyclicity in Law 9 we conclude that causal reasoning is of an abductive nature. Therefore, we propose to formalize causal reasoning in abductive causal theories defined as follows.

**Definition 3.1.1** (Abductive Causal Theory)**.** We define an **abductive causal theory** to be a tuple $\mathcal{T} := (\Delta, \mathfrak{A})$ consisting of a causal theory $\Delta$ and a set of formulas $\mathfrak{A}$, called **abducibles**, such that there is no causal rule $\phi \Rightarrow \psi \in \Delta$ and no abducible $\rho \in \mathfrak{A}$ with $\psi \models \rho$. We call $\mathcal{T}$ **atomic** or **literal** if $\Delta$ is an atomic or literal causal theory and if $\mathfrak{A}$ is a set of literals. In this case, a **pure** abducible $u \in \mathfrak{A}$ is a proposition with $\neg u \in \mathfrak{A}$.

Deviating from Bochman (10), we say that an atomic causal rule $b_1 \wedge ... \wedge b_n \Rightarrow h$ is **positive** if $b_1, ..., b_n, h \in \mathfrak{P}$ are propositions. A causal theory $\Delta$ is **positive** if it only mentions positive causal rules. Finally, we call an atomic abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ **positive** if $\Delta$ is a positive causal theory.

**Example 3.1.3.** If $\Delta$ is the causal theory consisting of Rules (3.3)-(3.6) from Example 3.1.1

$$\begin{aligned}
&\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob})\\
&\mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian})\\
&\mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob}),\\
&\mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{kilian})
\end{aligned}$$

we specify the abducibles

$$\mathfrak{A} := \{\mathit{spontaneously\_smokes}(X), \neg\mathit{spontaneously\_smokes}(X), \neg\mathit{smokes}(X)\}_{X \in \{\mathit{kilian}, \mathit{jakob}\}}$$

to obtain a positive abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$.

According to the principle of acyclicity in Law 9, every causal world of an abductive causal theory should be explained by its abducibles. This leads us to the following definition.

**Definition 3.1.2** (Exact Theories and Causal Worlds)**.** The **abductive closure** of an abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ is the causal theory $\Delta(\mathcal{T}) := \Delta \cup \{l \Rightarrow l : l \in \mathfrak{A}\}$. Further, the **consequence operator** $\mathcal{C}$ of $\mathcal{T}$ is the consequence operator of the abductive closure $\Delta(\mathcal{T})$ of $\mathcal{T}$. An **exact theory** $\Phi$ of $\mathcal{T}$ is an exact theory of the abductive closure $\Delta(\mathcal{T})$ that satisfies $\Phi = \mathcal{C}(\Phi \cap \mathfrak{A})$. Finally, a **causal world** $\omega$ of $\mathcal{T}$ is an exact theory that is a world. If $\Phi$ is an exact theory, the **(causal) explanation** $\epsilon := \epsilon(\Phi) := \Phi \cap \mathfrak{A}$ of $\Phi$ is the subset of all abducibles in $\Phi$.

**Example 3.1.4.** For the abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ of Example 3.1.3 where

$$\begin{aligned}\mathfrak{A} :=\ & \{\mathit{spontaneously\_smokes}(X), \neg \mathit{spontaneously\_smokes}(X), \neg \mathit{smokes}(X)\}_{X \in \{\mathit{kilian}, \mathit{jakob}\}}\\ \Delta :=\ & \{\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob}),\\ & \mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian}),\\ & \mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob}),\\ & \mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{kilian})\},\end{aligned}$$

we obtain the following causal theory of Example 3.1.1 as abductive closure.

$$\begin{aligned}& \mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{spontaneously\_smokes}(\mathit{jakob})\\ & \mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{spontaneously\_smokes}(\mathit{kilian})\\ & \mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob})\\ & \mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian})\\ & \mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob})\\ & \mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{kilian})\\ & \neg \mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \neg \mathit{spontaneously\_smokes}(\mathit{jakob})\\ & \neg \mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \neg \mathit{spontaneously\_smokes}(\mathit{kilian})\\ & \neg \mathit{smokes}(\mathit{jakob}) \Rightarrow \neg \mathit{smokes}(\mathit{jakob})\\ & \neg \mathit{smokes}(\mathit{kilian}) \Rightarrow \neg \mathit{smokes}(\mathit{kilian})\end{aligned}$$

Further, the causal worlds are given by

$$\begin{aligned}& \{\mathit{spontaneously\_smokes}(X), \mathit{smokes}(X) : X \in \{\mathit{kilian}, \mathit{jakob}\}\}\\ & \{\mathit{spontaneously\_smokes}(\mathit{jakob}), \mathit{smokes}(X) : X \in \{\mathit{kilian}, \mathit{jakob}\}\}\\ & \{\mathit{spontaneously\_smokes}(\mathit{kilian}), \mathit{smokes}(X) : X \in \{\mathit{kilian}, \mathit{jakob}\}\}\\ & \{\}.\end{aligned}$$

Fortunately, the causal worlds of an abductive causal theory can be characterized as follows.

**Proposition 3.1.1.** *Let $\mathcal{T} := (\Delta, \mathfrak{A})$ be a literal abductive causal theory, and let $\epsilon \subseteq \mathfrak{A}$ such that $\mathcal{C}(\epsilon)$ is a consistent set of propositional formulas. In this case, we find that $\mathcal{C}(\epsilon)$ is an exact theory of $\mathcal{T}$ with explanation $\epsilon$. In particular, a world $\omega$ is a causal world of $\mathcal{T}$ with explanation $\epsilon = \omega \cap \mathfrak{A}$ if and only if $\omega = \mathcal{C}(\epsilon)$.*

*Proof.* According to Theorem 2.3.2, $\epsilon \Rightarrow u$ cannot be derived from $\Delta(\mathcal{T})$ for any abducible $u \in \mathfrak{A} \setminus \epsilon$. Hence, we only need to show that $\mathcal{C}(\epsilon)$ is an exact theory of the abductive closure $\Delta(\mathcal{T})$, i.e.

$$\mathcal{C}(\mathcal{C}(\epsilon)) = \mathcal{C}(\epsilon).$$

Since $\epsilon \subseteq \mathfrak{A}$ is a subset of the defaults in the abductive closure $\Delta(\mathcal{T})$, we find $\epsilon \subseteq \mathcal{C}(\epsilon)$, i.e., from the monotonicity of $\mathcal{C}$ and Theorem 2.3.1 we derive $\mathcal{C}(\mathcal{C}(\epsilon)) \subseteq \mathcal{C}(\epsilon) \subseteq \mathcal{C}(\mathcal{C}(\epsilon))$. □

As in Section 2.3, we restrict ourselves to reasoning on causal theories with default negation that pin down whole worlds exactly, i.e., that satisfy Laws 7 and 8. Let $\mathcal{T} := (\Delta, \mathfrak{A})$ be an abductive causal theory. According to Law 7, all worlds $\omega$ are either exact theories or falsified, meaning that $\bot \in \mathcal{C}(\omega \cap \mathfrak{A})$. Furthermore, the principle of default negation in Law 8 means that every proposition $p \in \mathfrak{P}$ is considered to be false if it cannot be explained by causal reasoning, i.e., $\neg p \in \mathfrak{A}$ is an abducible.

**Definition 3.1.3** (Default Negation)**.** The **negative completion** of an atomic abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ is given by $\mathcal{T}^{nc} := (\Delta, \mathfrak{A} \cup \{\neg p : p \in \mathfrak{P}\})$. We say that an abductive causal theory $\mathcal{T}$ has **default negation** if it is the negative completion of an atomic abductive causal theory. If $\mathcal{T}$ is also the negative completion of a positive abductive causal theory, we call $\mathcal{T}$ a **positive** abductive causal theory with default negation.

**Example 3.1.5.** The abductive causal theory $\mathcal{T} = (\Delta, \mathfrak{A})$ in Example 3.1.3 where

$$\begin{aligned}\mathfrak{A} :=\ & \{spontaneously\_smokes(X), \neg spontaneously\_smokes(X), \neg smokes(X)\}_{X \in \{kilian, jakob\}}\\ \Delta :=\ & \{spontaneously\_smokes(jakob) \Rightarrow smokes(jakob),\\ & spontaneously\_smokes(kilian) \Rightarrow smokes(kilian),\\ & smokes(kilian) \Rightarrow smokes(jakob),\\ & smokes(jakob) \Rightarrow smokes(kilian)\},\end{aligned}$$

is derived as the negative completion of the atomic abductive causal theory that consists of the causal theory $\Delta$ and of the abducibles

$$\{spontaneously\_smokes(jakob), spontaneously\_smokes(kilian)\}.$$

Abductive causal theories with default negation can be used as a language for the causal models of Pearl (41) introduced in Section 2.3.1.

**Definition 3.1.4** (Bochman Transformation)**.** The **Bochman transformation** of a Boolean causal model $\mathcal{M} := (\mathbf{U}, \mathbf{V}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F})$ is the negative completion $\mathcal{T}(\mathcal{M})$ of the abductive causal theory that contains the causal rule $F_V(\mathrm{pa}(V), \mathrm{error}(V)) \Rightarrow V$ for every internal variable $V \in \mathbf{V}$ and has the abducibles $\mathbf{U}$.

**Example 3.1.6.** Let $\mathcal{M} := (\mathbf{U}, \mathbf{V}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F})$ be the causal model of Example 2.3.3.

$$\mathbf{U} := \quad \{sensor, rain\}$$

$$\mathcal{M}: \quad sprinkler := sensor \qquad wet := rain \lor sprinkler \qquad slippery := wet$$

In this case, the Bochman transformation $\mathcal{T}(\mathcal{M})$ is the abductive causal theory that consists of the causal theory

$$\Delta := \quad \{sensor \Rightarrow sprinkler, \qquad rain \lor sprinkler \Rightarrow wet, \qquad wet \Rightarrow slippery\}$$

and the abducibles

$$\mathfrak{A} := \{sensor, rain, \neg sensor, \neg sprinkler, \neg rain, \neg wet, \neg slippery\}.$$

**Remark 3.1.1.** Assuming that the functions $F_V(\mathrm{pa}(V), \mathrm{error}(V))$ are in disjunctive normal form and applying (Or) of Definition 2.3.7, we see that the Bochman transformation $\mathcal{T}(\mathcal{M})$ can be translated to an abductive causal theory with default negation while preserving the causal worlds.

**Example 3.1.7.** We can translate the causal theory $\Delta$ in Example 3.1.6 to the atomic theory

$$\begin{aligned}& sensor \Rightarrow sprinkler\\ & rain \Rightarrow wet\\ & sprinkler \Rightarrow wet\\ & wet \Rightarrow slippery.\end{aligned}$$

As desired, the causal worlds $\omega$ produced by the Bochman transformation $\mathcal{T}(\mathcal{M})$ of a causal model $\mathcal{M}$ correspond to solutions of $\mathcal{M}$.

**Theorem 3.1.2.** *If $\mathcal{M}$ is a Boolean causal model, every causal world $\omega$ of the Bochman transformation $\mathcal{T}(\mathcal{M})$ yields a solution of $\mathcal{M}$.*

*Proof.* This result is a direct consequence of Theorem 2.3.2, as every causal world of $\mathcal{T}(\mathcal{M})$ is a model of the completion of the corresponding abductive closure. □

Through the Bochman transformation, abductive causal theories define the feasible solutions of Boolean causal models that align with the Laws 1, 2, 5 – 9. Furthermore, the Bochman transformation canonically extends the treatment of external interventions from causal models to literal abductive causal theories.

**Definition 3.1.5** (Intervention)**.** Let $\mathcal{T} := (\Delta, \mathfrak{A})$ be a literal abductive causal theory, and let $\mathbf{i}$ be a truth value assignment on a set of propositions $\mathbf{I} \subseteq \mathfrak{P}$, not containing pure abducibles. To represent the intervention of forcing $\mathbf{i}$, we build the **modified theory** $\mathcal{T}_{\mathbf{i}} := (\Delta_{\mathbf{i}}, \mathfrak{A})$ that results from $\mathcal{T}$ by replacing all rules $\phi \Rightarrow l$ where the underlying proposition $p \in \mathfrak{P}$ of the literal $l \in \{p, \neg p\}$ lies in $\mathbf{I}$, i.e., $p \in \mathbf{I}$, with the rules $\top \Rightarrow l$, $l \in \mathbf{i}$.

**Example 3.1.8.** Recall the abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ in Example 3.1.3 where

$$\begin{aligned}
\mathfrak{A} :=\ & \{\mathit{spontaneously\_smokes}(X), \neg \mathit{spontaneously\_smokes}(X), \neg \mathit{smokes}(X)\}_{X \in \{\mathit{kilian}, \mathit{jakob}\}} \\
\Delta :=\ & \{\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob}), \\
& \mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian}), \\
& \mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob}), \\
& \mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{kilian})\}.
\end{aligned}$$

Assume we are Kilian and start smoking because we are influenced by Richard, another friend not modeled in $\mathcal{T} := (\Delta, \mathfrak{A})$. Hence, we need to intervene in $\mathcal{T}$ according to $\mathbf{i} := \{\mathit{smokes}(\mathit{kilian})\}$ and obtain the modified theory $\mathcal{T}_{\mathbf{i}} := (\Delta_{\mathbf{i}}, \mathfrak{A})$, where

$$\begin{aligned}
\mathfrak{A} :=\ & \{\mathit{spontaneously\_smokes}(X), \neg \mathit{spontaneously\_smokes}(X), \neg \mathit{smokes}(X)\}_{X \in \{\mathit{kilian}, \mathit{jakob}\}} \\
\Delta_{\mathbf{i}} :=\ & \{\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob}), \\
& \top \Rightarrow \mathit{smokes}(\mathit{kilian}), \\
& \mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob})\}.
\end{aligned}$$

As expected, the concept of intervention, defined in Definition 3.1.5, behaves consistently with the Bochman transformation.

**Proposition 3.1.3.** *For any Boolean causal model $\mathcal{M} := (\boldsymbol{U}, \boldsymbol{V}, \mathrm{Error}, \mathrm{Pa}, \boldsymbol{F})$ and any truth value assignment $\boldsymbol{i}$ on the internal variables $\boldsymbol{I} \subseteq \boldsymbol{V}$, we find that $\mathcal{T}(\mathcal{M}_{\boldsymbol{i}})$ and $\mathcal{T}(\mathcal{M})_{\boldsymbol{i}}$ have the same causal worlds.*

*Proof.* This result stems from the observation that the Bochman transformation, defined in Definition 3.1.4, essentially replaces the ":=" symbol with the "⇒" symbol. □

As announced at the beginning of this section, we now see how abductive causal theories with default negation correspond to abductive logic programming under the stable model semantics.

**Definition 3.1.6** (Bochman Transformation)**.** The **Bochman transformation** of an atomic abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ with default negation is defined to be the abductive logic program $\mathcal{P}(\mathcal{T}) := (\mathbf{P}, \mathfrak{A} \cap \mathfrak{P}, \emptyset)$, where

$$\mathbf{P} := \{p \leftarrow \phi |\ \phi \Rightarrow p \in \Delta \text{ atomic causal rule}\}.$$

**Example 3.1.9.** Recall the abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ in Example 3.1.3 where

$$\begin{aligned}\mathfrak{A} := \;& \{\mathit{spontaneously\_smokes}(X), \neg \mathit{spontaneously\_smokes}(X), \neg \mathit{smokes}(X)\}_{X \in \{kilian, jakob\}} \\ \Delta := \;& \{\mathit{spontaneously\_smokes}(jakob) \Rightarrow \mathit{smokes}(jakob), \\ & \mathit{spontaneously\_smokes}(kilian) \Rightarrow \mathit{smokes}(kilian), \\ & \mathit{smokes}(kilian) \Rightarrow \mathit{smokes}(jakob), \\ & \mathit{smokes}(jakob) \Rightarrow \mathit{smokes}(kilian)\}.\end{aligned}$$

The Bochman transformation of $\mathcal{T}$ is the abductive logic program $\mathcal{P}(\mathcal{T}) := (\mathbf{P}, \mathfrak{A}, \emptyset)$ consisting of the logic program

$$\begin{aligned}\mathbf{P}: \quad & \mathit{smokes}(jakob) \leftarrow \mathit{spontaneously\_smokes}(jakob) \\ & \mathit{smokes}(kilian) \leftarrow \mathit{spontaneously\_smokes}(kilian) \\ & \mathit{smokes}(jakob) \leftarrow \mathit{smokes}(kilian) \\ & \mathit{smokes}(kilian) \leftarrow \mathit{smokes}(jakob)\end{aligned}$$

and the abducibles $\mathfrak{A} := \{\mathit{spontaneously\_smokes}(X) : X \in \{jakob, kilian\}\}$.

To prove that the causal worlds of an abductive causal theory with default negation $\mathcal{T}$ correspond to stable models of the Bochman transformation $\mathcal{P}(\mathcal{T})$, we investigate the causal world semantics of abductive causal theories with default negation in greater detail.

**Definition 3.1.7** (Reduct)**.** Let $\mathcal{T} := (\Delta, \mathfrak{A})$ be an abductive causal theory with default negation, and let $\omega \subseteq \mathfrak{P}$ be a world. We define the **reduct** $\mathcal{T}_\omega := (\Delta_\omega, \mathfrak{A})$ to be the positive abductive causal theory with default negation obtained by first erasing all atomic causal rules $b_1 \wedge ... \wedge b_n \Rightarrow p$ from $\Delta$ if there is a negative literal $b_i := \neg p_i$, $1 \leq i \leq n$ with $\omega \models p_i$, and then erasing all negative literals in the causes of the remaining rules. If $\mathcal{P} := (\mathbf{P}, \mathfrak{A}, \mathbf{IC})$ is an abductive logic program, we define its **reduct** by $\mathcal{P}_\omega := (\mathbf{P}_\omega, \mathfrak{A}, \mathbf{IC})$.

As intended, the Bochman transformation respects our notion of reduct in causal logic.

**Lemma 3.1.4.** *For every abductive causal theory with default negation $\mathcal{T}$ and every world $\omega$ we have*

$$\mathcal{P}(\mathcal{T}_\omega) = \mathcal{P}(\mathcal{T})_\omega. \; \square$$

Further, the notion of a reduct is also well-behaved with the causal world semantics.

**Lemma 3.1.5.** *Let $\mathcal{T}$ be an abductive causal theory with default negation and let $\omega$ be a world. In this case, $\omega$ is a causal world of $\mathcal{T}$ if and only if it is a causal world of the reduct $\mathcal{T}_\omega$.*

*Proof.* Assume $\omega$ is a causal world of the theory $\mathcal{T}$ with explanation $\epsilon := \omega \cap \mathfrak{A}$. According to Proposition 3.1.1 and Theorem 2.3.3, we find for every literal $l$ that $l \in \omega$ if and only if $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T})$ and the following rules for sets of literals $\epsilon, \epsilon'$ and literals $l, l'$:

If $\epsilon \Rightarrow l$, then $\epsilon \cup \epsilon' \Rightarrow l$. **(Literal Monotonicity)**

If $\epsilon' \Rightarrow l$ and $\epsilon \cup \{l\} \Rightarrow l'$, then $\epsilon \cup \epsilon' \Rightarrow l'$. **(Literal Cut)**

$\{p, \neg p\} \Rightarrow l'$ for all positive literals $p$ and lierals $l'$ **(Literal Contradiction)**

We want to show that $\omega$ is a causal world of the reduct $\mathcal{T}_\omega$. This is the case if for every literal $l$ we find $l \in \omega$ if and only if we can derive $\epsilon \Rightarrow l$ from $\Delta(\mathcal{T}_\omega)$ and these axioms.

Let $l$ be a literal. Assume $l \in \omega$, that is, there is a proof of $\epsilon \Rightarrow l$ from $\Delta(\mathcal{T})$. We can derive every atomic rule $b_1 \wedge ... \wedge b_n \Rightarrow q \in \Delta(\mathcal{T})$ for which $\omega \models b_1, ..., b_m$ by (Strengthening) in Definition 2.3.5 from $\Delta(\mathcal{T}_\omega)$. We further observe that rules of the form $b_1 \wedge ... \wedge b_m \Rightarrow q$ for which $\omega \not\models b_i$ for some $i$ cannot occur in the proof of $\epsilon \Rightarrow l$. Indeed, assume without loss of generality that $i = 1$. Then every subsequent formula below $b_1 \wedge ... \wedge b_m \Rightarrow q$ in the proof tree would be of the form $b_1 \wedge c_1 \wedge ... \wedge c_k \Rightarrow p$ since (Literal Monotonicity) merely adds literals to the causes and $b_1$ cound only be removed by (Literal Cut) if $\epsilon \Rightarrow b_1$ would be derivable from $\Delta(\mathcal{T})$. However, in this case $b_1 \in \omega$, which is a contradiction to $\omega$ being a world. Note that with the same argument we find that (Literal Contradiction) cannot be applied in a proof of $\epsilon \Rightarrow l$.

Now assume $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T}_\omega)$. We want to show that $l \in \omega$, that is, $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T})$. To this end, we observe that without loss of generality every positive rule in $p_1 \wedge ... \wedge p_n \Rightarrow q \in \Delta(\mathcal{T}_\omega)$ corresponds to a rule in $\neg p'_1 \wedge ... \wedge \neg p'_m \wedge p_1 \wedge ... \wedge p_n \Rightarrow q \in \Delta(\mathcal{T})$ for positive literals $p_1, ..., p_n, p'_1, ..., p'_m$ such that $\omega \not\models p'_i$ for all $1 \leq i \leq m$. Thus, $p'_i \notin \omega$ and $\neg p'_i \in \omega$ as $\omega$ is a world. This means that $\epsilon \Rightarrow \neg p'_i$ is derivable from $\Delta(\mathcal{T})$ as and thus applying (Literal Cut), we find $\epsilon \cup \{p_1, ..., p_n\} \Rightarrow q$ can be deriven from $\Delta(\mathcal{T})$. Hence, we can derive $\epsilon \Rightarrow l$ from $\Delta(\mathcal{T})$.

Next, assume that $\omega$ is a causal world of the reduct $\mathcal{T}_\omega$ with explanation $\epsilon := \omega \cap \mathfrak{A}$, that is, for every literal $l$ we find that $l \in \omega$ if and only if $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T}_\omega)$ and the aforementioned axioms. We want to show that $\omega$ is a causal world of $\mathcal{T}$, that is, for a literal $l$ we have $l \in \omega$ if and only if $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T})$.

Let $l$ be a literal. Assume $l \in \omega$, i.e., $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T}_\omega)$. So as previously, take an atomic rule $b_1 \wedge ... \wedge b_n \Rightarrow q \in \Delta(\mathcal{T})$ with $\omega \models b_i$ for all negative literals $b_i$, $1 \leq i \leq n$. In this case, we find that $\epsilon \Rightarrow b_i$ is derivable from $\Delta(\mathcal{T})$ as $b_i \in \mathfrak{A}$ and therefore $b_i \in \epsilon$. We conclude with (Literal Cut) that $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T})$.

Finally, assume that $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T})$. We want to show that $l \in \omega$, i.e., that $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T}_\omega)$. First, all rules in $\Delta(\mathcal{T}_\omega)$ correspond to rules in $\Delta(\mathcal{T})$ by (Strengthening) in Definition 2.3.5. Let $b_1 \wedge ... \wedge b_n \Rightarrow l \in \Delta(\mathcal{T})$ be a rule that cannot be derived by (Strengthening) from $\Delta(\mathcal{T}_\omega)$. In this case, we may without loss of generality assume that $b_1 := \neg p_1$ is a negative literal with $\omega \models p_1$. Hence, we find that $\epsilon \Rightarrow p_1$ is derivable from $\Delta(\mathcal{T}_\omega)$. Further, by (Literal Contradiction) and (Literal Monotonicity), we find that $\{p_1, b_1, ..., b_n\} \Rightarrow p$ is derivable from $\Delta(\mathcal{T}_\omega)$. Hence, we apply (Literal Cut) and obtain $\epsilon \cup \{b_1, ..., b_n\} \Rightarrow p$. In summary, we conclude that $\epsilon \Rightarrow l$ is derivable from $\Delta(\mathcal{T}_\omega)$ and $l \in \omega$ as desired. □

Lastly, the Bochman transformation behaves well on abductive causal theories that correspond to positive programs.

**Lemma 3.1.6.** *Let $\mathcal{T} := (\Delta, \mathfrak{A})$ be a positive abductive causal theory with default negation, and let $\mathcal{P} := \mathcal{P}(\mathcal{T}) := (\boldsymbol{P}, \mathfrak{A}, \emptyset)$ be the Bochman transformation of $\mathcal{T}$. In this case, every stable model of $\mathcal{P}(\mathcal{T})$ is also a causal world of $\mathcal{T}$, and vice versa.*

*Proof.* Let $\omega$ be a causal world of $\mathcal{T} := (\Delta, \mathfrak{A})$ with explanation $\epsilon := \mathfrak{A} \cap \omega$. In this case, Theorem 2.3.2 ensures that $\omega$ is a supported model of $\mathbf{P}$. Furthermore, according to Theorem 2.3.3, for a positive literal $p$ we find $p \in \omega$ if and only if $\epsilon \Rightarrow p$ is derivable from $\Delta(\mathcal{T})$ using (Literal Monotonicity) and (Literal Cut). As literal monotonicity merely adds literals to the causes of rules, we conclude that we have a proof of $\epsilon \Rightarrow p$ only using (Literal Cut), which is equivalent to a proof of $p$ in terms of the immediate consequence operator of $\mathbf{P} \cup (\epsilon \cap \mathfrak{A})$.

Now, assume $\omega$ is a stable model of $\mathcal{P}$. Again, we have a proof for every proposition $p \in \omega$ in terms of the immediate consequence operator of $\mathbf{P} \cup \epsilon$, i.e., $\epsilon \Rightarrow p$ is derivable from $\Delta(\mathcal{T})$ using (Literal Cut). Hence, $\omega$ is a causal world of $\mathcal{T}$ as desired. □

Combining Lemma 3.1.4, Lemma 3.1.5, and Lemma 3.1.6 yields the main result of this section.

**Theorem 3.1.7.** *The Bochman transformation is a bijection, translating every abductive causal theory with default negation $\mathcal{T}$ to an abductive logic program $\mathcal{P}(\mathcal{T})$ such that the causal worlds of $\mathcal{T}$ are exactly the stable models of $\mathcal{P}(\mathcal{T})$.* □

Theorem 3.1.7 establishes the equivalence between abductive causal theories with default negation and general abductive logic programs without integrity constraints, underscoring their versatility in addressing diverse scenarios. Abductive logic programming under the stable model semantics encompasses various forms of causal reasoning, including those described by Laws 1, 2, and 5 – 9. This equivalence also holds for all types of abductive logic programs, i.e. acyclic, positive, stratified programs, etc., highlighting the broad applicability of our analysis in providing a causal interpretation for general abductive logic programs.

### The Logical and Explanatory Content of Causal Reasoning

As Bochman (10), we also observe that causal reasoning as formalized in abductive causal theories can be separated into logical and explanatory components.

**Definition 3.1.8** (Logical and Explanatory Content)**.** The **logical content** of a rule $R := (\phi \Rightarrow \psi)$ is the implication $\mathrm{logic}(R) := (\phi \rightarrow \psi)$. For a causal theory $\Delta$, the **logical content** is defined as $\mathrm{logic}(\Delta) := \{\mathrm{logic}(R) : R \in \Delta\}$. The **explanatory content** of $\Delta$ for a world $\omega$ is the causal theory $\Delta|_\omega := \{R \in \Delta : \omega \models \mathrm{logic}(R)\}$. If $\mathcal{T} := (\Delta, \mathfrak{A})$ is an abductive causal theory, the **logical content** is given by $\mathrm{logic}(\mathcal{T}) := \mathrm{logic}(\Delta)$, and the **explanatory content** is $\mathcal{T}|_\omega := (\Delta|_\omega, \mathfrak{A})$. In this case, $\mathcal{C}|_\omega$ denotes the corresponding consequence operator.

**Example 3.1.10.** Recall the abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ in Example 3.1.3 where

$$\begin{aligned}
\mathfrak{A} :=\ & \{\mathit{spontaneously\_smokes}(X), \neg\mathit{spontaneously\_smokes}(X), \neg\mathit{smokes}(X)\}_{X \in \{kilian, jakob\}} \\
\Delta :=\ & \{\mathit{spontaneously\_smokes}(jakob) \Rightarrow \mathit{smokes}(jakob), \\
& \mathit{spontaneously\_smokes}(kilian) \Rightarrow \mathit{smokes}(kilian), \\
& \mathit{smokes}(kilian) \Rightarrow \mathit{smokes}(jakob), \\
& \mathit{smokes}(jakob) \Rightarrow \mathit{smokes}(kilian)\}.
\end{aligned}$$

Further, let

$$\omega := \{\mathit{spontaneously\_smokes}(kilian), \mathit{smokes}(kilian)\}$$

be the world where Kilian (spontaneously) smokes, while Jakob is not smoking (spontaneously). The explanatory content $\mathcal{T}|_\omega$ is obtained by removing the rule $\mathit{smokes}(kilian) \Rightarrow \mathit{smokes}(jakob)$ from $\Delta$. The logical content, denoted as $\mathrm{logic}(\mathcal{T})$, is obtained by replacing the "$\Rightarrow$" symbol in $\Delta$ with the implication "$\rightarrow$".

The explanatory content of a causal theory $\Delta$ allows us to define the following event.

**Definition 3.1.9** (Decidability)**.** Let $\mathcal{T} := (\Delta, \mathfrak{A})$ be an abductive causal theory, and let $\omega$ be a world. We say that a formula $\phi$ is **(causally) decidable** in $\omega$ and write $\omega \models \mathrm{decides}(\phi)$, if

$$\phi \in \mathcal{C}|_\omega(\omega \cap \mathfrak{A}) \qquad \text{or} \qquad \neg\phi \in \mathcal{C}|_\omega(\omega \cap \mathfrak{A}).$$

We call $\omega$ a **logical world** of $\mathcal{T}$ and write $\mathrm{logic}_{\mathcal{T}}(\omega) = \mathrm{logic}(\omega)$ if $\omega \models \mathrm{logic}(\mathcal{T})$ and we call $\omega$ an **explainable world** of $\mathcal{T}$ if $\omega$ causally decides all formulas $\phi$, that is, $\omega \models \mathrm{decides}(\phi)$ for all formulas $\phi$.

We define the event $\mathrm{logic}(\mathcal{T})$ that $\mathcal{T}$ is **logical** to be the set of all logical worlds $\omega$, i.e.,

$$\mathrm{logic}(\mathcal{T}) := \{\omega \text{ world: } \omega \models \mathrm{logic}(\mathcal{T})\}.$$

Finally, we define the event that $\mathcal{T}$ **explains everything** to be the set of all explainable worlds, i.e.,

$$\mathrm{explains}(\mathcal{T}) := \{\omega \text{ world: } \omega \models \mathrm{decides}(\phi) \text{ for all formulas } \phi\}.$$

**Example 3.1.11.** Recall the abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ in Example 3.1.3 where

$$\begin{aligned}
\mathfrak{A} :=\ & \{\mathit{spontaneously\_smokes}(X), \neg\mathit{spontaneously\_smokes}(X), \neg\mathit{smokes}(X)\}_{X \in \{kilian, jakob\}} \\
\Delta :=\ & \{\mathit{spontaneously\_smokes}(jakob) \Rightarrow \mathit{smokes}(jakob), \\
& \mathit{spontaneously\_smokes}(kilian) \Rightarrow \mathit{smokes}(kilian), \\
& \mathit{smokes}(kilian) \Rightarrow \mathit{smokes}(jakob), \\
& \mathit{smokes}(jakob) \Rightarrow \mathit{smokes}(kilian)\}.
\end{aligned}$$

The theory $\mathcal{T}$ decides every formula in the world

$$\omega := \{\mathit{spontaneously\_smokes}(\mathit{kilian}), \mathit{smokes}(\mathit{kilian})\}$$

of Example 3.1.10. Hence, $\omega$ is an explainable world even though it is not a causal world.

As needed, the causal worlds of abductive causal theories can be characterized as follows.

**Proposition 3.1.8.** *A world $\omega$ is a causal world of an abductive causal theory $\mathcal{T}$ if and only if it is logical and explainable, i.e.,* $\mathrm{Causal}(\mathcal{T}) := \mathrm{logic}(\mathcal{T}) \cap \mathrm{explains}(\mathcal{T})$.

*Proof.* Assume $\omega$ is a causal world of $\mathcal{T}$. In this case, Theorem 2.3.2 yields $\omega \models \mathrm{logic}(\mathcal{T})$, i.e., $\omega$ is a logical world of $\mathcal{T}$. In particular, we find $\mathcal{T}|_\omega = \mathcal{T}$ and every formula $\phi$ is explainable since $\omega$ is an exact theory that is a world. Hence, $\omega$ is an explainable world.

Next, assume $\omega$ is a world that is logical and explainable with respect to $\mathcal{T}$. Hence, we find $\omega \models \mathrm{logic}(\mathcal{T})$ and $\mathcal{C}|_\omega(\omega \cap \mathfrak{A}) = \omega$. Again, we find $\mathcal{T}|_\omega = \mathcal{T}$ and $\omega$ is a causal world of $\mathcal{T}$. □

### Non-Ground Causal Reasoning

We aim to extend (abductive) causal theories to capture relations in a given domain, providing causal interpretations for non-ground abductive logic programs. To define semantics through grounding, we represent these relations using quantifier-free formulas $\phi$ in the fragment of relational first-order logic, which allows no quantifiers, no function symbols, and only finitely many constant and predicate symbols. These expressions are combined to form causal rules of the form $\phi \Rightarrow \psi$. Similarly to the propositional case, we do not allow nested expressions of the form $\phi \Rightarrow \psi \Rightarrow \rho \Rightarrow ....$ A finite set of causal rules constitutes a causal theory. Similarly to non-ground logic programs, we proceed by grounding to obtain the corresponding semantics of these causal theories.

**Remark 3.1.2.** Informally, a causal rule $\phi \Rightarrow \psi$ could also be interpreted as $\forall_{Y_1} ... \forall_{Y_n} \phi \Rightarrow \psi$, where $\mathrm{var}(\phi) \cup \mathrm{var}(\psi) := \{Y_1, ..., Y_n\}$. This interpretation arises from the fact that the Herbrand models of the corresponding universally closed formulas in prenex normal form can be obtained by grounding.

Let $\mathfrak{A}$ be a relational first-order alphabet. A **causal rule** is an expression $R := (\phi \Rightarrow \psi)$ for quantifier-free formulas $\phi$ and $\psi$. For a ground substitution $\gamma$ we define the corresponding **ground instance** to be $R^\gamma := (\phi^\gamma \Rightarrow \psi^\gamma)$. Furthermore, a **causal theory** $\Delta$ is a finite set of causal rules, and its **grounding** $\bar{\Delta}$ is defined as the set of all ground instances of rules in $\Delta$. An **exact theory** in the sense of Bochman (10) is then an exact theory of the grounding $\bar{\Delta}$. Finally, the causal theory $\Delta$ has **default negation** if its grounding $\bar{\Delta}$ has.

**Example 3.1.12.** Let $\mathfrak{A} := (\mathcal{C}, \mathcal{P})$ be the relational alphabet with constants $\mathcal{C} := \{\mathit{jakob}, \mathit{kilian}\}$ and predicates $\mathcal{P} := \{\mathit{smokes}, \mathit{spontaneously\_smokes}\}$, where each predicate has arity one. We can consider the following casual theory $\Delta$.

$$\begin{aligned}
&\mathit{spontaneously\_smokes}(X) \Rightarrow \mathit{spontaneously\_smokes}(X)\\
&\mathit{spontaneously\_smokes}(X) \Rightarrow \mathit{smokes}(X)\\
&\mathit{smokes}(X) \Rightarrow \mathit{smokes}(Y)\\
&\neg \mathit{spontaneously\_smokes}(X) \Rightarrow \neg \mathit{spontaneously\_smokes}(X)\\
&\neg \mathit{smokes}(X) \Rightarrow \neg \mathit{smokes}(X)
\end{aligned}$$

Further, we obtain the grounding $\bar{\Delta}$ below.

$$\begin{aligned}
&\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{spontaneously\_smokes}(\mathit{jakob})\\
&\mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{spontaneously\_smokes}(\mathit{kilian})\\
&\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob})\\
&\mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian})\\
&\mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob})\\
&\mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob})\\
&\mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{kilian})\\
&\mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian})\\
&\neg\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \neg\mathit{spontaneously\_smokes}(\mathit{jakob})\\
&\neg\mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \neg\mathit{spontaneously\_smokes}(\mathit{kilian})\\
&\neg\mathit{smokes}(\mathit{jakob}) \Rightarrow \neg\mathit{smokes}(\mathit{jakob})\\
&\neg\mathit{smokes}(\mathit{kilian}) \Rightarrow \neg\mathit{smokes}(\mathit{kilian})
\end{aligned}$$

Hence, as in Example 3.1.1, we obtain that

$$\omega := \{\mathit{smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{kilian})\},$$

is a causal world of $\Delta$.

Similarly to the propositional case, we extend non-ground causal theories with specified prior knowledge $\mathfrak{A}$, which serves as the starting point for explanations. This again results in abductive causal theories. To define the semantics through grounding, we again represent this prior knowledge using quantifier-free formulas in relational first-order logic.

Next, an abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{Ab})$ is a pair of a causal theory $\Delta$ and a set of quantifier-free formulas $\mathfrak{Ab}$, called **abducibles** such that the **grounding** $\bar{\mathcal{T}} := (\bar{\Delta}, \bar{\mathfrak{Ab}})$ is an abductive causal theory. A **causal world** $\omega$ of $\mathcal{T}$ is then a causal world of the grounding $\bar{\mathcal{T}}$. Finally, we say that $\mathcal{T}$ has **default negation** if its grounding $\bar{\mathcal{T}}$ has.

**Example 3.1.13.** In the alphabet $\mathfrak{A}$ of Example 3.1.12 we can consider the abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{Ab})$, consisting of the causal theory

$$\begin{aligned}
\Delta : \qquad & \mathit{spontaneously\_smokes}(X) \Rightarrow \mathit{smokes}(X)\\
& \mathit{smokes}(X) \Rightarrow \mathit{smokes}(Y)
\end{aligned}$$

and the abducibles

$$\mathfrak{Ab} := \{\mathit{spontaneously\_smokes}(X), \neg\mathit{spontaneously\_smokes}(X), \neg\mathit{smokes}(X)\}.$$

Finally, the causal worlds of $\mathcal{T}$ are the following worlds of Example 3.1.4.

$$\begin{aligned}
&\{\mathit{spontaneously\_smokes}(X), \mathit{smokes}(X) : X \in \{\mathit{kilian}, \mathit{jakob}\}\}\\
&\{\mathit{spontaneously\_smokes}(\mathit{jakob}), \mathit{smokes}(X) : X \in \{\mathit{kilian}, \mathit{jakob}\}\}\\
&\{\mathit{spontaneously\_smokes}(\mathit{kilian}), \mathit{smokes}(X) : X \in \{\mathit{kilian}, \mathit{jakob}\}\}\\
&\{\}
\end{aligned}$$

As the semantics of (abductive) causal theories and (abductive) logic programs is defined by grounding, Corollary 2.3.6 and Theorem 3.1.7 remain valid in the non-ground case.

## 3.2 Weighted Causal Reasoning

As the LogLinear models of Richardson and Domingos (46) extend propositional logic by uncertainty, we now extend literal abductive causal theories by unnormalized degrees of certainty. Hereby, we relate the uncertainty to the logical content while the explanations remain deterministic. Hence, as Bochman (10), we still consider causal reasoning as a binary relation on propositional formulas while extending the underlying logical content by degrees of certainty. Furthermore, we distinguish between the uncertainty in the abducibles, i.e., in the input data, and the uncertainty about the causal rules in our theory. Fix a propositional alphabet $\mathfrak{P}$.

**Definition 3.2.1** (Weighted Abductive Causal Theory)**.** A **weighted causal rule** $(w, R)$ consists of a weight $w \in \mathbb{R} \cup \{\infty\}$ and a literal causal rule $R$. A **weighted causal theory** $\Delta$ then is a finite set of weighted causal rules. Finally, the **explanatory content** of a weighted causal theory is the causal theory $\mathrm{EXP}(\Delta) := \{R : \exists_w (w, R) \in \Delta\}$.

A **weighted abductive causal theory** $\mathbb{T} := (\Delta, \mathfrak{A}, \Phi)$ consists of a weighted causal theory $\Delta$, a set of literals $\mathfrak{A}$, called **abducibles** and an **a priori knowledge**, which is a LogLinear model $\mathrm{PRIOR}(\mathbb{T}) := \Phi$ in the **pure** abducibles, i.e., in all positive literals $p \in \mathfrak{A}$ with $\neg p \in \mathfrak{A}$. The **constraint part** of $\mathbb{T}$ is the LogLinear model

$$\mathrm{CONST}(\mathbb{T}) := \{(w, \mathrm{logic}(R)) : (w, R) \in \Delta\}$$

and the **explanatory part** of $\mathbb{T}$ is the abductive causal theory

$$\mathrm{EXP}(\mathbb{T}) := (\mathrm{EXP}(\Delta), \mathfrak{A}).$$

**Remark 3.2.1.** We restrict ourselves to considering only literal causal rules in weighted abductive causal theories. In particular, we do not permit constraints, as their semantics within weighted abductive logic programming does no longer coincide with observing, i.e., conditioning on events.

**Example 3.2.1.** Assume that we are uncertain whether Jakob and Kilian influence each other to smoke in the abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ of Example 3.1.1.

$$\begin{aligned}
\mathfrak{A} :=\ & \{\mathit{spontaneously\_smokes}(X), \neg \mathit{spontaneously\_smokes}(X), \neg \mathit{smokes}(X)\}_{X \in \{\mathit{kilian}, \mathit{jakob}\}} \\
\Delta :=\ & \{\mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob}), \\
& \mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian}), \\
& \mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob}), \\
& \mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{kilian})\}
\end{aligned}$$

Further, assume they both independently start to smoke spontaneously, with a probability of 0.2. We may model the whole situation with the weighted abductive causal theory $\mathbb{T} := (\Delta, \mathfrak{A}, \Phi)$ consisting of the weighted causal theory $\Delta$, given by

$$\begin{gathered}
(\infty, \mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob})) \\
(\infty, \mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian}) \\
(\ln(2), \mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob})) \\
(\ln(2), \mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{kilian})),
\end{gathered}$$

the abducibles

$$\mathfrak{A} := \{\mathit{spontaneously\_smokes}(X), \neg \mathit{spontaneously\_smokes}(X), \neg \mathit{smokes}(X) \colon X \in \{\mathit{jakob}, \mathit{kilian}\}\}$$

and the a priori knowledge

$$\Phi := \{(\ln(0.2), \mathit{spontaneously\_smokes}(X)), (\ln(0.8), \neg \mathit{spontaneously\_smokes}(X))\}_{X \in \{\mathit{jakob}, \mathit{kilian}\}}.$$

To obtain a semantics for weighted abductive causal theories $\mathbb{T}$, we employ Parameterization 4 used in the LogLinear models of Section 2.2.2. However, this time, similar to the P-log semantics, we also account for the explanations provided by the explanatory part $\mathrm{EXP}(\mathbb{T})$, applying the principle of indifference in Law 3:

Informally, we interpret a weighted abductive causal theory $\mathbb{T} := (\Delta, \mathfrak{A}, \Phi)$ as a two-step procedure. First, the LogLinear model $\Phi$ describes a random experiment in which we choose a possible explanation $\epsilon$, that is, self-evident factual knowledge about the world outside the model that gives rise to a set of possible Aristotelian knowledge

$$\Omega|\epsilon := \{\omega \supseteq \epsilon \text{ explainable world of } \mathrm{EXP}(\mathbb{T})\}.$$

Further, the weights in $\Delta$ correspond to weak constraints in $\mathrm{CONST}(\mathbb{T})$ and express our degree of belief in a concrete world $\omega \in \Omega|\epsilon$. In particular, they express our degree of belief that $\omega$ is a logical world. Finally, if $\Omega|\epsilon = \emptyset$, we conclude that the factual knowledge $\epsilon$ is inconsistent with causal reasoning and conclude that $\epsilon$ cannot be observed. In particular, causal reasoning provides additional factual knowledge about the world outside the model, that is, $\epsilon$ must be consistent with causal reasoning.

**Definition 3.2.2** (Semantics of Weighted Abductive Causal Theories)**.** Let $\mathbb{T} := (\Delta, \mathfrak{A}, \Phi)$ be a weighted abductive causal theory and let $\omega$ be a world. The **possible explanation** of $\omega$ is the subset $\epsilon \subseteq \mathfrak{A}$ of all pure abducibles true in $\omega$. A formula $\phi$ is **(causally) decidable** in $\omega$, written $\omega \models \mathrm{decides}(\phi)$, if $\phi$ is decidable in $\omega$ with respect to the explanatory content $\mathrm{EXP}(\mathbb{T})$. We further say that $\omega$ is an **explainable world** if $\omega$ causally decides all formulas $\phi$, that is, $\omega \models \mathrm{decides}(\phi)$ for all formulas $\phi$. The event that $\mathbb{T}$ **explains everything** $\mathrm{explains}(\mathbb{T})$ is then given by the set of all explainable worlds.

We call a subset of pure abducibles $\epsilon \subseteq \mathfrak{A}$ a **consistent explanation** if there exists a an explainable world $\omega$ with possible explanation $\epsilon$ that satisfies all constraints with infinite weight in $\mathrm{CONST}(\mathbb{T})$, i.e., $\omega \models \phi$ whenever $(\infty, \phi) \in \mathrm{CONST}(\mathbb{T})$. In this case, we call $\epsilon$ the **explanation** of $\omega$. We define the event that the weighted abductive causal theory $\mathbb{T}$ is **consistent** by

$$\mathrm{consistent}(\mathbb{T}) := \{\epsilon \subseteq \mathfrak{A} \text{ consistent explanation}\}.$$

Finally, the **explanatory semantics** of $\mathbb{T}$ is the distribution assigning to every world $\omega$ with possible explanation $\epsilon$ the probability

$$\pi(\omega) := \pi_{\mathbb{T}}(\omega) := \pi_{\mathbb{T}}^{exp}(\omega) := \pi_{\mathrm{CONST}(\mathbb{T})}(\omega|\epsilon, \mathrm{explains}(\mathbb{T})) \cdot \pi_{\mathrm{PRIOR}(\mathbb{T})}(\epsilon|\, \mathrm{consistent}(\mathbb{T})).$$

Here, we identify every possible explanation $\epsilon$ with the event $\{\omega \text{ world: } \omega \supseteq \epsilon\}$.

Again, the probability of a formula $\phi$ is defined to be

$$\pi(\phi) := \pi_{\mathbb{T}}(\phi) := \pi_{\mathbb{T}}^{exp}(\phi) := \sum_{\substack{\omega \text{ world} \\ \omega \models \phi}} \pi(\omega).$$

**Example 3.2.2.** Recall the weighted abductive causal theory $\mathbb{T} := (\Delta, \mathfrak{A}, \Phi)$ in Example 3.2.1.

$$\begin{aligned}
\mathfrak{A} :=\ & \{spontaneously\_smokes(X), \neg spontaneously\_smokes(X), \neg smokes(X)\}_{X \in \{kilian, jakob\}} \\
\Phi :=\ & \{(\ln(0.2), spontaneously\_smokes(jakob)), \\
& (\ln(0.8), \neg spontaneously\_smokes(jakob)), \\
& (\ln(0.2), spontaneously\_smokes(kilian)), \\
& (\ln(0.8), \neg spontaneously\_smokes(kilian))\} \\
\Delta :=\ & \{(\infty, spontaneously\_smokes(jakob) \Rightarrow smokes(jakob)) \\
& (\infty, spontaneously\_smokes(kilian) \Rightarrow smokes(kilian) \\
& (\ln(2), smokes(kilian) \Rightarrow smokes(jakob)) \\
& (\ln(2), smokes(jakob) \Rightarrow smokes(kilian))\}
\end{aligned}$$

Denoting *spontaneously_smokes*(_) by *s_smokes*(_), the explanatory part EXP($\mathbb{T}$) has the following explainable worlds with non-zero weight.

$$\begin{aligned}
& \omega(\{\}) = 4 \\
& \omega(\{s\_smokes(kilian), smokes(kilian)\}) = 2 \\
& \omega(\{s\_smokes(kilian), smokes(kilian), smokes(jakob)\}) = 4 \\
& \omega(\{s\_smokes(jakob), smokes(jakob)\}) = 2 \\
& \omega(\{s\_smokes(jakob), smokes(jakob), smokes(kilian)\}) = 4 \\
& \omega(\{s\_smokes(kilian), s\_smokes(jakob), smokes(jakob), smokes(kilian)\}) = 4
\end{aligned}$$

We observe $\pi_\Phi(\text{consistent}(\mathbb{T})) = 1$. Hence, Kilian or Jakob smokes with probability

$$\begin{aligned}
& \pi_{\mathbb{T}}^{exp}(smokes(kilian)) = \\
& = 1 \cdot \pi_\Phi(s\_smokes(kilian)) + \frac{2}{3} \cdot \pi_\Phi(s\_smokes(jakob), \neg s\_smokes(kilian)) = \\
& = \frac{1}{5} + \frac{2}{3} \cdot \frac{1}{5} \cdot \frac{4}{5} = \frac{23}{75} \\
& \pi_{\mathbb{T}}^{exp}(smokes(jakob)) = \pi_{\mathbb{T}}^{exp}(smokes(kilian)) = \frac{23}{75}.
\end{aligned}$$

Kilian and Jakob smoke with probability

$$\pi_{\mathbb{T}}^{exp}(smokes(jakob), smokes(kilian)) = 1 \cdot \frac{1}{25} + \frac{2}{3} \cdot \frac{4}{25} + \frac{2}{3} \cdot \frac{4}{25} = \frac{19}{75}.$$

Consequently, if we observe that Jakob is smoking, Kilian smokes with a probability of

$$\pi(smokes(kilian) | smokes(jakob)) = \frac{19}{23}.$$

Let us add the weighted causal rule $(\infty, \neg \mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob}))$ stating that Jakob smokes resulting in the following weighted abductive causal theory $\mathbb{T}'$.

$$\mathfrak{A}' := \{\mathit{spontaneously\_smokes}(X), \neg\mathit{spontaneously\_smokes}(X), \neg\mathit{smokes}(X)\}_{X \in \{\mathit{kilian}, \mathit{jakob}\}}$$

$$\begin{aligned}\Phi' := \; & \{(-\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{jakob})),\\ & (-\ln(0.8), \neg\mathit{spontaneously\_smokes}(\mathit{jakob})),\\ & (-\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{kilian})),\\ & (-\ln(0.8), \neg\mathit{spontaneously\_smokes}(\mathit{kilian}))\}\end{aligned}$$

$$\begin{aligned}\Delta' := \; & \{(\infty, \mathit{spontaneously\_smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob}))\\ & (\infty, \mathit{spontaneously\_smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{kilian})\\ & (\ln(2), \mathit{smokes}(\mathit{kilian}) \Rightarrow \mathit{smokes}(\mathit{jakob}))\\ & (\ln(2), \mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{kilian})),\\ & (\infty, \neg\mathit{smokes}(\mathit{jakob}) \Rightarrow \mathit{smokes}(\mathit{jakob}))\}\end{aligned}$$

The explanatory part $\mathrm{EXP}(\mathbb{T}')$ has the following explainable worlds with non-zero weight.

$$\begin{aligned}&\omega(\{\mathit{s\_smokes}(\mathit{kilian}), \mathit{smokes}(\mathit{kilian}), \mathit{smokes}(\mathit{jakob})\}) = 4\\ &\omega(\{\mathit{s\_smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{jakob})\}) = 2\\ &\omega(\{\mathit{s\_smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{kilian})\}) = 4\\ &\omega(\{\mathit{s\_smokes}(\mathit{kilian}), \mathit{s\_smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{kilian})\}) = 4\end{aligned}$$

The event that $\mathbb{T}'$ is consistent is given by

$$\mathrm{consistent}(\mathbb{T}') := \{\{\mathit{s\_smokes}(\mathit{jakob})\}, \{\mathit{s\_smokes}(\mathit{kilian})\}, \{\mathit{s\_smokes}(\mathit{jakob}), \mathit{s\_smokes}(\mathit{kilian})\}\},$$

that is, $\pi_\Phi(\mathrm{consistent}(\mathbb{T}')) = 1 - \frac{16}{25} = \frac{9}{25}$.

Hence we find Kilian smoking with a probability

$$\pi_{\mathbb{T}'}(\mathit{smokes}(\mathit{kilian})) = \frac{25}{9} \cdot \left(1 \cdot \frac{1}{5} + \frac{2}{3} \cdot \frac{4}{25}\right) = \frac{23}{27} \neq \frac{19}{23} = \pi_{\mathbb{T}}(\mathit{smokes}(\mathit{kilian}) | \mathit{smokes}(\mathit{jakob}))$$

We conclude that adding constraints with infinite weight is not the same as observing these constraints.

Again, applying the principle of default negation in Law 8, we restrict ourselves to theories of the following type.

**Definition 3.2.3** (Default Negation). A weighted abductive causal theory $\mathbb{T}$ has **default negation** if its explanatory part $\mathrm{EXP}(\mathbb{T})$ has.

**Example 3.2.3.** The abductive causal theory $\mathbb{T} := (\Delta, \mathfrak{A}, \Phi)$ in Example 3.2.1 has a default negation.

$$\mathfrak{A} := \quad \{spontaneously\_smokes(X), \neg spontaneously\_smokes(X), \neg smokes(X)\}_{X \in \{kilian, jakob\}}$$

$$\begin{aligned}\Phi := \quad & \{(\ln(0.2), spontaneously\_smokes(jakob)),\\ & (\ln(0.8), \neg spontaneously\_smokes(jakob)),\\ & (\ln(0.2), spontaneously\_smokes(kilian)),\\ & (\ln(0.8), \neg spontaneously\_smokes(kilian))\}\end{aligned}$$

$$\begin{aligned}\Delta := \quad & \{(\infty, spontaneously\_smokes(jakob) \Rightarrow smokes(jakob))\\ & (\infty, spontaneously\_smokes(kilian) \Rightarrow smokes(kilian)\\ & (\ln(2), smokes(kilian) \Rightarrow smokes(jakob))\\ & (\ln(2), smokes(jakob) \Rightarrow smokes(kilian))\}\end{aligned}$$

Having formalized the causal reasoning under uncertainty given by the Laws 1 − 3, 5 − 9 and Parametrization 4, we now relate the resulting weighted abductive causal theories with default negation to already existing frameworks in statistical relational artificial intelligence.

**Definition 3.2.4** (Weighted Abductive Logic Program)**.** A **weighted abductive logic program** $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ consists of an LP$^{\text{MLN}}$ program $\mathbf{P}$, a set of **abducibles** $\mathfrak{A} \subseteq \mathfrak{P}$ and an **a priori knowledge**, i.e. a LogLinear model $\Phi$ in $\mathfrak{A}$ such that $(\text{LP}(\mathbf{P}), \mathfrak{A})$ is an abductive logic program. A **stable model** $\omega$ of $\mathbb{P}$ with **explanation** $\epsilon$ is a stable model of the abductive logic program $(\text{LP}(\mathbf{P}|_\omega), \mathfrak{A})$ that has the explanation $\epsilon$.

Finally, the **Bochman transformation** of a weighted abductive causal theory $\mathbb{T} := (\Delta, \mathfrak{A}, \Phi)$ with default negation is given by the weighted abductive logic program

$$\mathbb{P}(\mathbb{T}) := \big(\{(w, p \leftarrow \phi)\}_{(w, \phi \Rightarrow p) \in \Delta}, \mathfrak{A} \cap \mathfrak{P}, \Phi\big).$$

**Example 3.2.4.** Recall the weighted abductive causal theory $\mathbb{T} := (\Delta, \mathfrak{A}, \Phi)$ in Example 3.2.1.

$$\mathfrak{A} := \quad \{spontaneously\_smokes(X), \neg spontaneously\_smokes(X), \neg smokes(X)\}_{X \in \{kilian, jakob\}}$$

$$\begin{aligned}\Phi := \quad & \{(\ln(0.2), spontaneously\_smokes(jakob)),\\ & (\ln(0.8), \neg spontaneously\_smokes(jakob)),\\ & (\ln(0.2), spontaneously\_smokes(kilian)),\\ & (\ln(0.8), \neg spontaneously\_smokes(kilian))\}\end{aligned}$$

$$\begin{aligned}\Delta := \quad & \{(\infty, spontaneously\_smokes(jakob) \Rightarrow smokes(jakob))\\ & (\infty, spontaneously\_smokes(kilian) \Rightarrow smokes(kilian)\\ & (\ln(2), smokes(kilian) \Rightarrow smokes(jakob))\\ & (\ln(2), smokes(jakob) \Rightarrow smokes(kilian))\}\end{aligned}$$

The Bochman transformation of $\mathbb{T}$ is the weighted abductive logic program $\mathbb{P}(\mathbb{T}) := (\mathbf{P}, \mathfrak{A}, \Phi)$ given by the LP$^{\text{MLN}}$ program $\mathbf{P}$ with the clauses

$$\begin{aligned}& (\infty, smokes(jakob) \leftarrow spontaneously\_smokes(jakob))\\ & (\infty, smokes(kilian) \leftarrow spontaneously\_smokes(kilian))\\ & (\ln(2), smokes(jakob) \leftarrow smokes(kilian))\\ & (\ln(2), smokes(kilian) \leftarrow smokes(jakob)),\end{aligned}$$

the abducibles

$$\mathfrak{A} := \{\mathit{spontaneously\_smokes}(\mathit{jakob}), \mathit{spontaneously\_smokes}(\mathit{kilian})\}$$

and the a priori knowledge

$$\begin{aligned}\Phi := \quad & \{(\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{jakob})),\\ & (\ln(0.8), \neg \mathit{spontaneously\_smokes}(\mathit{jakob})),\\ & (\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{kilian})),\\ & (\ln(0.8), \neg \mathit{spontaneously\_smokes}(\mathit{kilian}))\}\end{aligned}$$

The Bochman transformation yields the following semantics for weighted abductive logic programs.

**Definition 3.2.5** (Semantics of Weighted Abductive Logic Programs)**.** Let $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ be a weighted abductive logic program. An $\mathfrak{A}$-structure $\epsilon \subseteq \mathfrak{A}$ is **consistent** with respect to $\mathbb{P}$ if there exists at least one stable model $\omega$ of $\mathbb{P}$ with explanation $\epsilon$ such that $\omega \models C$ for all clauses with infinite weight $(\infty, C) \in \mathbf{P}$, that is, $(\infty, C) \in \mathbf{P}|_{\omega}$. In this case, we also say that $\epsilon$ is **consistent**. We define the event of $\mathbb{P}$ being **consistent** by

$$\text{consistent}(\mathbb{P}) := \{\epsilon \subseteq \mathfrak{A}\colon\ \epsilon \text{ consistent } \mathfrak{A}\text{-structure}\}.$$

The **abductive closure** of $\mathbb{P}$ is defined to be the $\text{LP}^{\text{MLN}}$ program

$$\bar{\mathbf{P}} := \mathbf{P} \cup \{(0, u)\colon\ u \in \mathfrak{A}\}.$$

Further, we associate to a stable model $\omega$ of $\mathbb{P}$ with explanation $\epsilon$ the probability

$$\pi_{\mathbb{P}}(\omega) := \pi_{\bar{\mathbf{P}}}(\omega|\epsilon) \cdot \pi_{\Phi}(\epsilon|\, \text{consistent}(\mathbb{P})).$$

Here, we identify $\epsilon$ with the event $\{\omega$ stable model of $\mathbb{P}$ with explanation $\epsilon\}$. The probability of a formula $\phi$ is then defined by

$$\pi_{\mathbb{P}}(\phi) := \sum_{\substack{\omega \text{ stable model of } \mathbb{P}\\ \omega \models \phi}} \pi_{\mathbb{P}}(\omega).$$

As desired, the Bochman transformation is a bijection that behaves well regarding the explanatory semantics of weighted abductive causal theories with default negation.

**Lemma 3.2.1.** *Let $\mathbb{T} := (\Delta, \mathfrak{A}, \Phi)$ be a weighted abductive causal theory with default negation that has the Bochman transformation $\mathbb{P} := \mathbb{P}(\mathbb{T}) := (\boldsymbol{P}, \mathfrak{A} \cap \mathfrak{P}, \Phi)$. In this case, a world $\omega$ is a stable model of $\mathbb{P}$ with explanation $\epsilon = \omega \cap (\mathfrak{A} \cap \mathfrak{P})$ if and only if $\omega$ is an explainable world of $\mathbb{T}$ with explanation for $\epsilon$.*

*Proof.* As $\mathbb{T}$ has default negation, a pure abducible is a positive literal $p \in \mathfrak{A}$. Hence, the possible explanation $\epsilon$ of a world $\omega$ is the $\mathfrak{A} \cap \mathfrak{P}$-structure

$$\epsilon = \omega \cap (\mathfrak{A} \cap \mathfrak{P}).$$

The construction of $\mathbf{P}|_{\omega}$ in Definition 2.2.34 and the construction of $\text{EXP}(\mathbb{T})|_{\omega}$ in Definition 3.1.8 yields $(\text{LP}(\mathbf{P}|_{\omega}), \mathfrak{A}) = \mathcal{P}(\text{EXP}(\mathbb{T})|_{\omega})$. Furthermore, $\omega$ is an explainable world if and only if $\omega$ is a causal world of $\text{EXP}(\mathbb{T})|_{\omega}$. Therefore, the desired result follows from Theorem 3.1.7. □

**Theorem 3.2.2.** *The explanatory semantics of each weighted abductive causal theory* $\mathbb{T} = (\Delta, \mathfrak{A}, \Phi)$ *with default negation coincides with the semantics of its Bochman transformation*

$$\mathbb{P} := \mathbb{P}(\mathbb{T}) = (\boldsymbol{P}, \mathfrak{A} \cap \mathfrak{P}, \Phi).$$

*Proof.* Let $\omega$ be a stable model of the weighted abductive logic program $\mathbb{P}$ with explanation $\epsilon$. According to Lemma 3.2.1 this is the case if and only if $\omega$ is an explainable word with respect to $\mathbb{T}$ that has the possible explanation of $\epsilon$. Hence, we conclude that it is sufficient to show $\pi_{\mathbb{T}}(\omega) = \pi_{\mathbb{P}}(\omega)$. Recall Definition 3.2.2 and 3.2.5:

$$\begin{aligned} \pi_{\mathbb{T}}(\omega) &= \pi_{\mathrm{CONST}(\mathbb{T})}(\omega \mid \epsilon, \mathrm{explains}(\mathbb{T})) \cdot \pi_{\Phi}(\epsilon \mid \mathrm{consistent}(\mathbb{T})) \\ \pi_{\mathbb{P}}(\omega) &= \pi_{\bar{\mathbf{P}}}(\omega \mid \epsilon) \cdot \pi_{\Phi}(\epsilon \mid \mathrm{consistent}(\mathbb{P})) \end{aligned} \tag{3.7}$$

According to Lemma 3.2.1 an $(\mathfrak{A} \cap \mathfrak{P})$-structure $\epsilon'$ is consistent regarding $\mathbb{P}$ if and only if it is a consistent explanation regarding $\mathbb{T}$. Hence, we conclude consistent$(\mathbb{T})$ = consistent$(\mathbb{P})$ and

$$\pi_{\Phi}(\epsilon \mid \mathrm{consistent}(\mathbb{T})) = \pi_{\Phi}(\epsilon \mid \mathrm{consistent}(\mathbb{P})).$$

Observe that $\mathbf{P} = \mathrm{CONST}(\mathbb{T})$ and $\bar{\mathbf{P}} = \mathbf{P} \cup \{(0, u) \colon u \in \mathfrak{A} \cap \mathfrak{P}\}$. Let $\omega'$ be an arbitrary stable model of the abductive closure $\bar{\mathbf{P}}$, that is, $\omega'$ is a stable model of $\mathbb{P}$ that satisfies all clauses with infinite weight, i.e., $\omega' \models C$ whenever $(\infty, C) \in \mathbf{P}$. According to Lemma 3.2.1 this is the case if and only if $\omega'$ is an explainable world of $\mathbb{T}$ that satisfies all constraints with infinite weight in the constraint part CONST$(\mathbb{T})$, i.e. $\omega' \models C$ whenever $(\infty, C) \in \mathrm{CONST}(\mathbb{T})$. The construction of the semantics for LogLinear models and LP$^{\mathrm{MLN}}$ programs in Definition 2.2.11 and 2.2.34 then yields the following identities:

$$w_{\bar{\mathbf{P}}}(\omega') \overset{\text{Definition 2.2.34}}{=} \exp\left( \sum_{\substack{(w,C) \in \bar{P} \\ w \neq \infty \\ \omega' \models C}} w \right) = \exp\left( \sum_{\substack{(w,C) \in P \\ w \neq \infty \\ \omega' \models C}} w \right) \overset{\text{Definition 2.2.11}}{=} w_{\mathrm{CONST}(\mathbb{T})}(\omega')$$

$$w_{\bar{\mathbf{P}}}(\top) = \sum_{\substack{\omega'' \text{ stable} \\ \text{model of } \bar{\mathbf{P}}}} w_{\mathbf{P}}(\omega'') = \sum_{\substack{\omega'' \in \mathrm{explains}(\mathbb{T}) \\ \omega'' \models C \\ \text{for all } (\infty, C) \in \mathrm{CONST}(\mathbb{T})}} w_{\mathrm{CONST}(\mathbb{T})}(\omega'') = w_{\mathrm{CONST}(\mathbb{T})}(\mathrm{explains}(\mathbb{T}))$$

$$\pi_{\bar{\mathbf{P}}}(\omega') = \frac{w_{\bar{\mathbf{P}}}(\omega')}{w_{\bar{\mathbf{P}}}(\top)} = \frac{w_{\mathrm{CONST}(\mathbb{T})}(\omega')}{w_{\mathrm{CONST}(\mathbb{T})}(\mathrm{explains}(\mathbb{T}))} = \pi_{\mathrm{CONST}(\mathbb{T})}(\omega' \mid \mathrm{explains}(\mathbb{T})).$$

We conclude that the distributions $\pi_{\bar{\mathbf{P}}}(\_)$ and $\pi_{\mathrm{CONST}(\mathbb{T})}(\_ \mid \mathrm{explains}(\mathbb{T}))$ coincide, i.e.,

$$\pi_{\mathrm{CONST}(\mathbb{T})}(\omega \mid \epsilon, \mathrm{explains}(\mathbb{T})) = \pi_{\bar{\mathbf{P}}}(\omega \mid \epsilon).$$

Therefore, the desired result follows from (3.7). □

Weighted abductive logic programming also generalizes the reasoning provided by the probabilistic causal models of Pearl (41).

**Definition 3.2.6** (Bochman Transformation)**.** Let $\mathbb{M} := (\mathcal{M}, \pi)$ be a Boolean probabilistic causal model, where $\mathcal{M} := (\mathbf{U}, \mathbf{V}, \mathrm{Error}, \mathrm{Pa}, \mathbf{F})$. For every equation $V := F_V(\mathrm{pa}(V), \mathrm{error}(V))$ in $\mathcal{M}$ we identify the function $F_V(\mathrm{pa}(V), \mathrm{error}(V))$ with a formula that has disjunctive normal form $C_1^V \vee ... \vee C_n^V$. Now, let $\mathbf{P}$ be the LP$^{\mathrm{MLN}}$ program given by the rules $(\infty, V \leftarrow C_i^V)$ and let $\Phi$ be the LogLinear model consisting of a weak constraint $(\ln(\pi(\mathbf{u})), \mathbf{u})$ for every situation $\mathbf{u}$ of the causal model $\mathcal{M}$. The **Bochman transformation** of $\mathbb{M}$ then is the weighted abductive logic program $\mathbb{P}(\mathbb{M}) := (\mathbf{P}, \mathbf{U}, \Phi)$.

**Example 3.2.5.** Recall the probabilistic causal model $\mathbb{M} := (\mathcal{M}, \pi)$ in Example 2.3.3.

$$\begin{array}{lll} \mathcal{M}: & \mathit{cloudy} := u_1 & \\ & \mathit{rain} := \mathit{cloudy} \wedge u_2 & \mathit{sprinkler} := (\mathit{cloudy} \wedge u_3) \vee (\neg \mathit{cloudy} \wedge u_4) \\ & \mathit{wet} := (\mathit{rain} \vee \mathit{sprinkler}) \wedge u_5 & \mathit{slippery} := \mathit{wet} \wedge u_6 \end{array}$$

$u_i$ mutually independent Boolean random varibales

$$\begin{array}{ll} \pi(u_1) = 0.5 & \pi(u_2) = 0.6 \\ \pi(u_3) = 0.1 & \pi(u_4) = 0.7 \\ \pi(u_5) = 0.9 & \pi(u_6) = 0.8 \end{array}$$

The Bochman transformation $\mathbb{P}(\mathbb{M}) := (\mathbf{P}, \mathfrak{A}, \Phi)$ of $\mathbb{M}$ is given by the LP$^{\text{MLN}}$ program $\mathbf{P}$

$$\begin{array}{ll} (\infty, \mathit{cloudy} \leftarrow u_1) & \\ (\infty, \mathit{sprinkler} \leftarrow \mathit{cloudy}, u_2) & (\infty, \mathit{sprinkler} \leftarrow \neg \mathit{cloudy}, u_3) \\ (\infty, \mathit{rain} \leftarrow \mathit{cloudy}, u_4) & (\infty, \mathit{rain} \leftarrow \neg \mathit{cloudy}, u_5) \\ (\infty, \mathit{wet} \leftarrow \mathit{rain}, u_6) & (\infty, \mathit{wet} \leftarrow \mathit{sprinkler}, u_6) \\ (\infty, \mathit{slippery} \leftarrow \mathit{wet}, u_7) & \end{array}$$

the abducibles $\mathfrak{A} := \{u_1, ..., u_6\}$ and the prior knowledge $\Phi$

$$\begin{array}{l} (\ln(0.5 \cdot 0.6 \cdot 0.1 \cdot 0.7 \cdot 0.9 \cdot 0.8), u_1 \wedge u_2 \wedge u_3 \wedge u_4 \wedge u_5 \wedge u_6) \\ (\ln(0.5 \cdot 0.6 \cdot 0.1 \cdot 0.7 \cdot 0.9 \cdot 0.8), \neg u_1 \wedge u_2 \wedge u_3 \wedge u_4 \wedge u_5 \wedge u_6) \\ (\ln(0.5 \cdot 0.4 \cdot 0.1 \cdot 0.7 \cdot 0.9 \cdot 0.8), u_1 \wedge \neg u_2 \wedge u_3 \wedge u_4 \wedge u_5 \wedge u_6) \\ ... \\ (\ln(0.5 \cdot 0.4 \cdot 0.9 \cdot 0.3 \cdot 0.1 \cdot 0.2), \neg u_1 \wedge \neg u_2 \wedge \neg u_3 \wedge \neg u_4 \wedge \neg u_5 \wedge \neg u_6). \end{array}$$

Recall from Section 2.3.3 that causal models correspond to Bochman's causal theories and abductive logic programs under the supported model semantics. Hence, if the supported and stable model semantics coincide in the corresponding abductive logic program, the Bochman transformation $\mathbb{P}(\mathbb{M})$ of a probabilistic Boolean causal model $\mathbb{M}$ produces the same distribution as $\mathbb{M}$ itself.

**Proposition 3.2.3.** *Let $\mathbb{M} := (\mathcal{M}, \pi)$ be a probabilistic causal model with the Bochman transformation $\mathbb{P}(\mathbb{M}) := (\boldsymbol{P}, \boldsymbol{U}, \Phi)$. If every situation $\boldsymbol{u}$ of the causal model $\mathcal{M}$ gives rise to a $\boldsymbol{U}$-structure consistent with $\mathbb{P}$, the distribution induced by a probabilistic causal model $\mathbb{M}$ coincides with the semantics of its Bochman transformation $\mathbb{P}(\mathbb{M})$.*

*Proof.* Let $\mathbb{M} := (\mathcal{M}, \pi)$ be a Boolean probabilistic causal model with Bochman transformation $\mathbb{P}(\mathbb{M}) := (\mathbf{P}, \mathbf{U}, \Phi)$. In this case, we find that $\mathbb{M}$ yields a well-defined distribution only if every assignment of truth values $\mathbf{u}$ to the external variables $\mathbf{U}$ of the causal model $\mathcal{M}$ yields a unique solution $\omega(\mathbf{u})$, which is the unique supported model of $\text{LP}(\mathbf{P}) \cup \mathbf{u}$. Since $\mathbf{u}$ also gives rise to a consistent $\mathbf{U}$-structure with respect to $\mathbb{P}$, we find that $\text{LP}(\mathbf{P}) \cup \mathbf{u}$ has at least one stable model. Now, Theorem 1 of Gelfond and Lifschitz (27) yields that $\omega(\mathbf{u})$ is the unique stable model of $\text{LP}(\mathbf{P}) \cup \mathbf{u}$. Finally, the desired result follows from the construction of $\Phi$, the observation $\pi(\text{consistent}(\mathbb{P}(\mathbb{M}))) = 1$, and the fact that all the uncertainty of $\mathbb{P}(\mathbb{M})$ is concentrated in the a priori knowledge $\Phi$, i.e. $\pi_{\mathbb{P}(\mathbb{M})}(\omega(\mathbf{u})) = \pi_\Phi(\omega(\mathbf{u}) \cap \mathbf{U}) = \pi(\mathbf{u}) = \pi_{\mathbb{M}}(\omega(\mathbf{u}))$. □

The Bochman transformation $\mathbb{P}(\mathbb{M})$ assigns to a probabilistic causal model $\mathbb{M}$ a distribution, which is determined by Laws $1 - 3$, $5 - 9$. In addition, it induces the following notion of intervention in weighted abductive logic programs.

**Definition 3.2.7** (Intervention)**.** Let $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ be a weighted abductive logic program and let $\mathbf{i}$ be a truth value assignment on a set of propositions $\mathbf{I} \subseteq \mathfrak{P} \setminus \mathfrak{A}$, which does not contain any abducibles. To represent the intervention of enforcing $\mathbf{i}$, we construct the **modified program** $\mathbb{P}_\mathbf{i} := (\mathbf{P}_\mathbf{i}, \mathfrak{A}, \Phi)$, where $\mathbf{P}_\mathbf{i}$ is derived from $\mathbf{P}$ by removing all clauses $(w, C) \in \mathbf{P}$ with $\mathrm{head}(C) \in \mathbf{I}$ and adding a fact $(\infty, p)$ for all positive literals $p \in \mathbf{i}$.

**Example 3.2.6.** Recall the weighted abductive logic program $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ in Example 3.2.4.

$$\mathfrak{A} := \qquad \{\mathit{spontaneously\_smokes}(\mathit{jakob}), \mathit{spontaneously\_smokes}(\mathit{kilian})\}$$

$$\begin{array}{ll}\Phi := \qquad & \{(\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{jakob})),\\ & (\ln(0.8), \neg\mathit{spontaneously\_smokes}(\mathit{jakob})),\\ & (\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{kilian})),\\ & (\ln(0.8), \neg\mathit{spontaneously\_smokes}(\mathit{kilian}))\}\end{array}$$

$$\begin{array}{ll}\mathbf{P}: \qquad & (\infty, \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{spontaneously\_smokes}(\mathit{jakob}))\\ & (\infty, \mathit{smokes}(\mathit{kilian}) \leftarrow \mathit{spontaneously\_smokes}(\mathit{kilian}))\\ & (\ln(2), \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{smokes}(\mathit{kilian}))\\ & (\ln(2), \mathit{smokes}(\mathit{kilian}) \leftarrow \mathit{smokes}(\mathit{jakob}))\end{array}$$

If we apply the intervention $\mathbf{i} := \{\mathit{smokes}(\mathit{kilian})\}$ of Example 3.1.8 to the program $\mathbb{P} = (\mathbf{P}, \mathfrak{A}, \Phi)$ we obtain the modified program $\mathbb{P}_\mathbf{i} := (\mathbf{P}_\mathbf{i}, \mathfrak{A}, \Phi)$ resulting from $\mathbb{P}$ by replacing the $\mathrm{LP}^{\mathrm{MLN}}$ program $\mathbf{P}$ with the program

$$\begin{array}{ll}\mathbf{P}_\mathbf{i} \qquad & (\infty, \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{spontaneously\_smokes}(\mathit{jakob}))\\ & (\infty, \mathit{smokes}(\mathit{kilian}))\\ & (\ln(2), \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{smokes}(\mathit{kilian})).\end{array}$$

Therefore, if Kilian smokes because he is influenced by Richard, Jakob smokes with a probability

$$\pi(\mathit{smokes}(\mathit{jakob}) |\, \mathrm{do}(\mathit{smokes}(\mathit{kilian})) = 11/15.$$

As required, the notion of intervention in Definition 3.2.7 is consistent with the Bochman transformation.

**Proposition 3.2.4.** *Let $\mathbb{M} := (\mathcal{M}, \pi)$ be a probabilistic Boolean causal model, and $\boldsymbol{i}$ be a truth value assignment on the internal variables $\boldsymbol{I} \subseteq \boldsymbol{V}$. In this case, intervening before applying the Bochman transformation $\mathbb{P}(\mathbb{M}_\boldsymbol{i})$ and applying the Bochman transformation before intervening $\mathbb{P}(\mathbb{M})_\boldsymbol{i}$ yields the same distribution on $\mathfrak{P}$.*

*Proof.* This is proven analogously to Proposition 2.3.5. □

## 3.3 Counterfactual Reasoning

Finally, we transfer counterfactual reasoning from probabilistic causal models to weighted abductive logic programs. Hence, fix a weighted abductive logic program $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ together with two truth value assignments $\mathbf{e}, \mathbf{i}$ on sets of propositions $\mathbf{E}, \mathbf{I} \subseteq \mathfrak{P} \setminus \mathfrak{A}$, not containing any abducibles. We ask for the probability $\pi_\mathbb{P}(\phi | \mathbf{e}, \mathrm{do}(\mathbf{i}))$ that a formula $\phi$ would hold if we had enforced $\mathbf{i}$ before observing the evidence $\mathbf{e}$. According to Lewis (37), this means, we ask for the probability of $\phi$ to hold in an alternative world $\omega^a$, minimally deviating from the observed real world $\omega^r$ such that $\mathbf{i}$ becomes true.

**Example 3.3.1.** Recall the weighted abductive logic program $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ of Example 3.2.4.

$$\mathfrak{A} := \quad \{\mathit{spontaneously\_smokes}(\mathit{jakob}), \mathit{spontaneously\_smokes}(\mathit{kilian})\}$$

$$\begin{aligned} \Phi := \quad & \{(\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{jakob})), \\ & (\ln(0.8), \neg \mathit{spontaneously\_smokes}(\mathit{jakob})), \\ & (\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{kilian})), \\ & (\ln(0.8), \neg \mathit{spontaneously\_smokes}(\mathit{kilian}))\} \end{aligned}$$

$$\begin{aligned} \mathbf{P} : \quad & (\infty, \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{spontaneously\_smokes}(\mathit{jakob})) \\ & (\infty, \mathit{smokes}(\mathit{kilian}) \leftarrow \mathit{spontaneously\_smokes}(\mathit{kilian})) \\ & (\ln(2), \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{smokes}(\mathit{kilian})) \\ & (\ln(2), \mathit{smokes}(\mathit{kilian}) \leftarrow \mathit{smokes}(\mathit{jakob})) \end{aligned}$$

Hence, we consider two friends, Jakob and Kilian, which we denote by the constants *jakob* and *kilian*, respectively. Both friends may smoke, denoted by $\mathit{smokes}(\mathit{jakob})$ and $\mathit{smokes}(\mathit{kilian})$, because they start spontaneously to smoke, denoted by

$$\mathit{spontaneously\_smokes}(\mathit{jakob}) \text{ and } \mathit{spontaneously\_smokes}(\mathit{kilian}),$$

or because the other friend smokes.

In this example, we assume the observation that Jakob and Kilian smoke, i.e., we observe

$$\mathbf{e} := \{\mathit{smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{kilian})\}.$$

We now could ask for the probability

$$\pi_{\mathbb{P}}(\mathit{smokes}(\mathit{kilian}) | \mathit{smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{kilian}), \mathrm{do}(\neg \mathit{smokes}(\mathit{jakob})))$$

that Kilian would smoke if Jakob were not smoking, i.e. if we additionally consider the intervention

$$\mathbf{i} := \{\neg \mathit{smokes}(\mathit{jakob})\}.$$

As the evidence $\mathbf{e}$ and intervention $\mathbf{i}$ contradict each other, this is a counterfactual query. Hence, we are interested in the alternative worlds $\omega^a$ that satisfy our intervention $\mathbf{i}$ – where Jakob does not smoke – while minimally deviating from the observed real world $\omega^r$, where both Jakob and Kilian smoke.

To represent the alternative worlds $\omega^a$, we proceed like Balke and Pearl (5), i.e. we build modified twin programs:

First, we make two copies $\mathfrak{I}^{a/r}$ of the non-abducibles $\mathfrak{I} := \mathfrak{P} \setminus \mathfrak{A}$ and set $u^{a/r} = u$ for all abducibles $u \in \mathfrak{A}$, yielding maps $\_^{a/r}$ of literals, (weighted) clauses, truth value assignments, etc. Here, the superscripts $a/r$ denote the alternative and real worlds, indicating that everything unexplained remains unchanged in the alternative worlds. The **twin program** $\mathbb{P}^T := (\mathbf{P^T}, \mathfrak{A}, \Phi)$ is then obtained by replacing the LP$^{\mathrm{MLN}}$ program $\mathbf{P}$ in $\mathbb{P}$ with the program $\mathbf{P}^T := \mathbf{P}^a \cup \mathbf{P}^r$. Finally, to model the constraint $\mathbf{i}$ on the alternative world, we follow Pearl (41) and intervene in the twin program $\mathbb{P}^T$ according to $\mathbf{i}^a$, resulting in the **modified twin program** $\mathbb{P}^T_{\mathbf{i}^a} := (\mathbf{P}^a_{\mathbf{i}^a} \cup \mathbf{P}^r, \mathfrak{A}, \Phi)$.

**Example 3.3.2.** Recall the weighted abductive logic program $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ of Example 3.2.4.

$$\begin{array}{ll}
\mathfrak{A} := & \{\mathit{spontaneously\_smokes}(\mathit{jakob}), \mathit{spontaneously\_smokes}(\mathit{kilian})\} \\
\\
\Phi := & \{(\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{jakob})), \\
& (\ln(0.8), \neg \mathit{spontaneously\_smokes}(\mathit{jakob})), \\
& (\ln(0.2), \mathit{spontaneously\_smokes}(\mathit{kilian})), \\
& (\ln(0.8), \neg \mathit{spontaneously\_smokes}(\mathit{kilian}))\} \\
\\
\mathbf{P}: & (\infty, \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{spontaneously\_smokes}(\mathit{jakob})) \\
& (\infty, \mathit{smokes}(\mathit{kilian}) \leftarrow \mathit{spontaneously\_smokes}(\mathit{kilian})) \\
& (\ln(2), \mathit{smokes}(\mathit{jakob}) \leftarrow \mathit{smokes}(\mathit{kilian})) \\
& (\ln(2), \mathit{smokes}(\mathit{kilian}) \leftarrow \mathit{smokes}(\mathit{jakob}))
\end{array}$$

In Example 3.3.1, we are interested in the alternative worlds $\omega^a$ that satisfy the intervention

$$\mathbf{i} := \{\neg \mathit{smokes}(\mathit{jakob})\},$$

while minimally deviating from the observed real world $\omega^r$, where we observe

$$\mathbf{e} := \{\mathit{smokes}(\mathit{jakob}), \mathit{smokes}(\mathit{kilian})\}.$$

To represent these alternative worlds $\omega^a$, we first build the twin program $\mathbb{P}^T := (\mathbf{P}^T, \mathfrak{A}, \Phi)$ by replacing the LP$^{\text{MLN}}$ program $\mathbf{P}$ with the program $\mathbf{P}^T$ below, where we write $\mathit{s\_smokes}/1$ instead of $\mathit{spontaneously\_smokes}/1$.

$$\begin{array}{lll}
\mathbf{P}^T: & (\infty, \mathit{smokes}(\mathit{kilian})^r \leftarrow \mathit{s\_smokes}(\mathit{kilian})) & (\infty, \mathit{smokes}(\mathit{kilian})^a \leftarrow \mathit{s\_smokes}(\mathit{kilian})) \\
& (\infty, \mathit{smokes}(\mathit{jakob})^r \leftarrow \mathit{s\_smokes}(\mathit{jakob})) & (\infty, \mathit{smokes}(\mathit{jakob})^a \leftarrow \mathit{s\_smokes}(\mathit{jakob})) \\
& (\ln(2), \mathit{smokes}(\mathit{jakob})^r \leftarrow \mathit{smokes}(\mathit{kilian})^r) & (\ln(2), \mathit{smokes}(\mathit{jakob})^a \leftarrow \mathit{smokes}(\mathit{kilian})^a) \\
& (\ln(2), \mathit{smokes}(\mathit{kilian})^r \leftarrow \mathit{smokes}(\mathit{jakob})^r) & (\ln(2), \mathit{smokes}(\mathit{kilian})^a \leftarrow \mathit{smokes}(\mathit{jakob})^a)
\end{array}$$

Next, we model the constraint $\mathbf{i}$ on the alternative worlds and intervene according to $\mathbf{i}^a$, i.e. we further replace $\mathbf{P}^T$ with the following LP$^{\text{MLN}}$ program $\mathbf{P}^T_{\neg \mathit{smokes}(\mathit{jakob})^a}$, yielding the modified weighted abductive logic program $\mathbb{P}^T_{\mathbf{i}^a} := (\mathbf{P}^T_{\mathbf{i}^a}, \mathfrak{A}, \Phi)$.

$$\begin{array}{lll}
\mathbf{P}^T_{\mathbf{i}^a}: & (\infty, \mathit{smokes}(\mathit{kilian})^r \leftarrow \mathit{s\_smokes}(\mathit{kilian})) & (\infty, \mathit{smokes}(\mathit{kilian})^a \leftarrow \mathit{s\_smokes}(\mathit{kilian})) \\
& (\infty, \mathit{smokes}(\mathit{jakob})^r \leftarrow \mathit{s\_smokes}(\mathit{jakob})) & \\
& (\ln(2), \mathit{smokes}(\mathit{jakob})^r \leftarrow \mathit{smokes}(\mathit{kilian})^r) & \\
& (\ln(2), \mathit{smokes}(\mathit{kilian})^r \leftarrow \mathit{smokes}(\mathit{jakob})^r) & (\ln(2), \mathit{smokes}(\mathit{kilian})^a \leftarrow \mathit{smokes}(\mathit{jakob})^a)
\end{array}$$

However, if we have no constraint on the alternative world, i.e., if $\mathbf{i} = \emptyset$, the real and alternative world coincide, and we should recover the conditional probabilities, that is, we declare the following law to hold.

**Law 10** (Consistency of Counterfactual Reasoning)**.** *For every formula $\phi$ and for each assignment of truth values $\boldsymbol{e}$ on a subset of propositions $\boldsymbol{E} \subseteq \mathfrak{P} \setminus \mathfrak{A}$ we find $\pi_{\mathbb{P}}(\phi|\boldsymbol{e}) = \pi_{\mathbb{P}}(\phi|\boldsymbol{e}, \mathrm{do}(\emptyset))$.*

If we naively proceed like Balke and Pearl (5) and compute counterfactual probabilities by setting $\pi_{\mathbb{P}}(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i})) := \pi_{\mathbb{P}^T_{\mathbf{i}^a}}(\phi^a|\mathbf{e}^r)$, unfortunately we obtain results contradicting Law 10.

**Example 3.3.3.** In Example 3.3.2, we find that

$$\pi_{\mathbb{P}^T}(smokes(kilian)^a|(smokes(kilian)^r) = 41/44 \neq 1 = \pi_{\mathbb{P}}(smokes(kilian)|(smokes(kilian)).$$

Note that considering no intervention, that is, $\mathbf{i} := \emptyset$, we find $\mathbb{P}^T_{\mathbf{i}^a} = \mathbb{P}^T$.

**Example 3.3.4.** Assume Felix and Kailin are drinking tea while the bell rings. If Kailin or Felix hears the bell, one of them stands up to open the door while the other remains seated. Now we denote by $felix$ the event that Felix stands up to open the door and by $kailin$ the event that Kailin stands up to open the door. This could result in the weighted abductive logic program $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ which is given by the LP$^{\text{MLN}}$ program $\mathbf{P}$

$$(\infty, felix \leftarrow \neg kailin, u) \qquad\qquad (\infty, kailin \leftarrow \neg felix, v),$$

the abducibles $\mathfrak{A} := \{u, v\}$ and the a priori knowledge

$$\Phi := \{(-\ln(2), u), (-\ln(2), \neg u), (-\ln(2), v), (-\ln(2), \neg v)\}.$$

We then find the probability

$$\pi_{\mathbb{P}^T}(felix^a|felix^r) = 0.75 \neq 1 = \pi_{\mathbb{P}}(felix|felix).$$

Consider a consistent $\mathfrak{A}$-structure $\epsilon$. Further, let $\omega_1, ..., \omega_n$ be the stable models $\omega$ of $\mathbb{P}^T_{\mathbf{i}^a}$ with $\pi_{\mathbb{P}^T_{\mathbf{i}^a}}(\omega) > 0$ and explanation $\epsilon$. As the real and the alternative worlds $\omega_i \cap \mathfrak{P}^{a/r}$ share the same abducibles, we conclude from Examples 3.3.3 and 3.3.4 that the modified twin program $\mathbb{P}^T_{\mathbf{i}^a}$ fails to distribute the mass of the explanation $\pi_{\mathbb{P}^T_{\mathbf{i}^a}}(\epsilon|\mathbf{e}^r) = \pi_{\mathbb{P}}(\epsilon|\mathbf{e})$ to the alternative worlds $\omega_i \cap \mathfrak{P}^a$ correctly. In particular, the uncertainty in the LP$^{\text{MLN}}$ program $\mathbf{P}^a_{\mathbf{i}^a}$ lacks an explanation in terms of the abducibles $\mathfrak{A}$, making it infeasible to use the principle of indifference in Law 3, Parametrization 4 and $\mathbf{P}^a_{\mathbf{i}^a}$ for distributing the mass $\pi_{\mathbb{P}}(\epsilon|\mathbf{e})$ to the alternative worlds $\omega_i \cap \mathfrak{P}^a$. However, there is no reason to doubt in the alternative worlds $\omega_i \cap \mathfrak{P}^a$ or the hard constraints of $\mathbf{P}^a_{\mathbf{i}^a}$. Therefore, we propose to estimate the counterfactual probability $\pi_{\mathbb{P}}(\phi|\mathbf{e}, \text{do}(\mathbf{i}))$ as follows:

Intervene in $\mathbb{P}$ according to $\mathbf{i}$ and obtain the modified program $\mathbb{P}_{\mathbf{i}} := (\mathbf{P}_{\mathbf{i}}, \mathfrak{A}, \Phi)$. Since the stable models and hard constraints of the LP$^{\text{MLN}}$ program $\mathbf{P}^a_{\mathbf{i}^a} = \mathbf{P}_{\mathbf{i}}$ stay correct regarding the alternative worlds $\omega_i \cap \mathfrak{P}^a$, we propose the estimate $l \leq \pi(\phi|\mathbf{e}, \text{do}(\mathbf{i})) \leq u$, where

$$l := \sum_{\substack{\epsilon\ \mathfrak{A}\text{-structure} \\ \pi_{\bar{\mathbf{P}}_{\mathbf{i}}}(\phi|\epsilon)=1}} \pi_{\mathbb{P}}(\epsilon|\mathbf{e}, \text{consistent}(\mathbb{P}_{\mathbf{i}})) \text{ and } u := \sum_{\substack{\epsilon\ \mathfrak{A}\text{-structure} \\ \pi_{\bar{\mathbf{P}}_{\mathbf{i}}}(\phi|\epsilon)>0}} \pi_{\mathbb{P}}(\epsilon|\mathbf{e}, \text{consistent}(\mathbb{P}_{\mathbf{i}})). \tag{3.8}$$

Here, $\bar{\mathbf{P}}_{\mathbf{i}}$ denotes the abductive closure of the program $\mathbb{P}_{\mathbf{i}}$. Hence, like the Credal Semantics of Cozman and Mauá (18; 17), we propose to answer counterfactual queries on general weighted abductive logic programs with a range of possible values, i.e. $\pi_{\mathbb{P}}(\phi|\mathbf{e}, \text{do}(\mathbf{i})) \in [l, u]$.

**Remark 3.3.1.** If we find for the evidence $\mathbf{e}$ that $\pi(\mathbf{e}) = 0$, we obtain $\pi(\phi|\mathbf{e}, \text{do}(\mathbf{i})) = 0$ for every assignment of the truth value $\mathbf{i}$ on propositions $\mathbf{I} \subseteq \mathfrak{P} \setminus \mathfrak{A}$.

**Example 3.3.5.** If we apply our procedure to Examples 3.3.3 and 3.3.4, we obtain the statements $\pi(smokes(kilian)|smokes(kilian), \text{do}(\emptyset)) \in [0.75, 1]$ and $\pi(felix|felix, \text{do}(\emptyset)) \in [0.5, 1]$, aligned with Law 10.

**Example 3.3.6.** In the case of Example 3.3.2, we obtain a sharp probability

$$\pi(smokes(kilian)|smokes(jakob), smokes(kilian), \text{do}(\neg smokes(jakob))) = \frac{11}{19}$$

that Kilian would smoke if Jakob were not smoking. If we observe, however, that nobody is smoking, we only obtain the trivial statement that Kilian would smoke if Jakob were smoking with a probability

$$\pi(smokes(kilian)|\neg smokes(jakob), \neg smokes(kilian), \text{do}(smokes(jakob))) \in [0, 1].$$

Recall that we interpret a weighted abductive logic program $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ as a two-step procedure. The LogLinear model $\Phi$ describes our factual knowledge of the world outside the model as a random experiment where we choose a possible explanation $\epsilon$ leading to the possible worlds $\Omega|\epsilon := \{\omega \text{ stable model of } \mathbb{P} \text{ with explanation } \epsilon\}$. Further, the weights in $\mathbf{P}$ correspond to weak constraints in the constraint part of a weighted abductive causal theory and express our degree of belief in choosing a concrete world $\omega \in \Omega|\epsilon$. However, the selection of this concrete world $\omega \in \Omega|\epsilon$ remains unexplained by the program $\mathbb{P}$.

To derive a counterfactual probability $\pi_{\mathbb{P}}(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$, we use evidence $\mathbf{e}$ to learn about the possible explanation $\epsilon$ of the possible worlds in $\Omega|\epsilon$, i.e., we replace the factual knowledge about the world outside our model encoded in $\Phi$ with the distribution $\pi_{\mathbb{P}}(\_|\mathbf{e})$. In addition, we replace the LP$^{\text{MLN}}$ program $\mathbf{P}$ by the modified program $\mathbf{P}_{\mathbf{i}}$ to ensure the constraint $\mathbf{i}$ on the alternative worlds. As the selection of a concrete alternative world

$$\omega^a \in \Omega^a|\epsilon := \{\omega \text{ stable model of } \mathbb{P}_{\mathbf{i}} \text{ with explanation } \epsilon\}$$

is not explained by the program $\mathbb{P}$, we can only estimate the probability $\pi_{\mathbb{P}}(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$:

If $\pi_{\bar{\mathbf{P}}_{\mathbf{i}}}(\phi|\epsilon) = 1$, we necessarily choose an alternative world that satisfies the query $\phi$. In particular, knowing about $\epsilon$ already enforces the query $\phi$ in the alternative world, that is, $\pi_{\mathbb{P}}(\epsilon|\mathbf{e})$ contributes to the lower bound $l$. If $\pi_{\bar{\mathbf{P}}_{\mathbf{i}}}(\phi|\epsilon) > 0$, we have the possibility to choose a world that satisfies the query. In particular, knowing about $\epsilon$ could lead to an alternative world satisfying the query $\phi$, i.e., $\pi_{\mathbb{P}}(\epsilon|\mathbf{e})$ contributes to the upper bound $u$. Finally, we account for information about the world outside the model provided by the causal explanation of the alternative world. By posing a counterfactual query, we a priori assume that the state of the world outside the model is not inconsistent with the causal description of the alternative world. Hence, we only consider explanations consistent with our causal knowledge encoded in $\mathbb{P}_{\mathbf{i}}$ and condition on consistent$(\mathbb{P}_{\mathbf{i}})$ in the definition of the upper bound $u$ and the lower bound $l$.

**Remark 3.3.2.** If one does not want to assume that the state of the world outside the model is consistent with the causal description of the alternative world encoded in $\mathbb{P}_{\mathbf{i}}$, one could multiply $u$ and $l$ with $\pi_{\mathbb{P}}(\text{consistent}(\mathbb{P}_{\mathbf{i}}))$ to obtain the corresponding lower and upper bounds of the counterfactual probability $\pi_{\mathbb{P}}(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$.

## 3.4 The Causal Interpretation of Statistical Relational Artificial Intelligence

In this section, we adopt the viewpoint that weighted abductive logic programming is the most general formalism for causal reasoning, emerging from Laws $1-3$, $5-9$ and Parametrization 4. We now study its relationship with other formalisms in statistical relational artificial intelligence. However, before doing so, we transfer the results of Section 3.2 to the non-ground case.

### Non-Ground Weighted Causal Reasoning

We aim to extend weighted abductive causal theories to capture relations in a given domain, providing causal interpretations for formalisms in statistical relational artificial intelligence. As in the case of Markov logic networks (46) in Section 2.2.3, to define a semantics via grounding, we represent these relations using quantifier-free formulas $\phi$ in the fragment of relational first-order logic, which allows no quantifiers, no function symbols, and only finitely many constant and predicate symbols. These expressions are combined to weighted constraints $(w, \phi)$ or weighted causal rules of the form $(w, \phi \Rightarrow \psi)$ for $w \in \mathbb{R}$. Similarly to the propositional case, we do not allow nested expressions of the form $\phi \Rightarrow \psi \Rightarrow \rho \Rightarrow ....$ A weighted abductive causal theory is a triple $\mathbb{T} := (\Delta, \mathfrak{Ab}, \Phi)$, where $\Delta$ is a finite set of weighted causal rules, $\mathfrak{Ab}$ is a set of literals and $\Phi$ is a Markov logic network in the pure abducibles, that is, in all positive literals $p \in \mathfrak{Ab}$ with $\neg p \in \mathfrak{Ab}$. Similarly to non-ground LP$^{\text{MLN}}$ programs (35), we proceed by grounding to obtain the corresponding semantics.

Let $\mathfrak{A}$ be a relational first-order alphabet. A **weighted causal rule** $(w, R)$ consists of a weight $w \in \mathbb{R} \cup \{\infty\}$ and a literal causal rule $R$. For a ground substitution $\gamma$ the corresponding **ground instance** is given by $(w, R^{\gamma})$. Further, a **weighted causal theory** $\Delta$ is a finite set of weighted causal rules, and the **grounding** $\bar{\Delta}$ of $\Delta$ is given by the set of all ground instances of weighted rules in $\Delta$. Finally, a **weighted abductive causal theory** $\mathbb{T} := (\Delta, \mathfrak{Ab}, \Phi)$ consists of a weighted causal theory $\Delta$, a set of literals $\mathfrak{Ab}$ called **abducibles**, and a Markov logic network $\Phi$ in the **pure abducibles**, i.e., it only mentions atoms $p \in \mathfrak{Ab}$ with $\neg p \in \mathfrak{Ab}$. The **grounding** of $\mathbb{T}$ is defined as $\bar{\mathbb{T}} := (\bar{\Delta}, \bar{\mathfrak{Ab}}, \bar{\Phi})$; we say that the theory $\mathbb{T}$ has **default negation** if its grounding $\bar{\mathbb{T}}$ has and we associate with $\mathbb{T}$ the distribution $\pi_{\mathbb{T}}(\_)$, which is the explanatory semantics of the grounding $\bar{\mathbb{T}}$.

**Example 3.4.1.** Recall the abductive causal theory $\mathcal{T} := (\Delta, \mathfrak{A})$ in Example 3.2.1.

$$\mathfrak{Ab} := \quad \{\mathit{spontaneously\_smokes}(X), \neg \mathit{spontaneously\_smokes}(X), \neg \mathit{smokes}(X)\}$$

$$\begin{aligned} \Delta := \quad & \{\mathit{spontaneously\_smokes}(X) \Rightarrow \mathit{smokes}(X), \\ & \mathit{smokes}(X) \Rightarrow \mathit{smokes}(Y)\} \end{aligned}$$

Assume we are unsure whether Kilian and Jakob influence each other to smoke and assume that everyone smokes spontaneously, independently, with a probability of 0.2. This could result in the weighted abductive causal theory $\mathbb{T} := (\Delta, \mathfrak{Ab}, \Phi)$ which consists of the weighted causal theory

$$\begin{aligned} \Delta := \quad & \{(\infty, \mathit{spontaneously\_smokes}(X) \Rightarrow \mathit{smokes}(X)), \\ & (\ln(2), \mathit{smokes}(X) \Rightarrow \mathit{smokes}(Y))\} \end{aligned}$$

the abducibles

$$\mathfrak{Ab} := \quad \{\mathit{spontaneously\_smokes}(X), \neg \mathit{spontaneously\_smokes}(X), \neg \mathit{smokes}(X)\},$$

and the Markov logic network

$$\Phi := \quad \{(\ln(0.2), \mathit{spontaneously\_smokes}(X)), (\ln(0.8), \neg \mathit{spontaneously\_smokes}(X))\}.$$

The semantics of $\mathbb{T}$ is given by the semantics of the propositional weighted abductive causal theory $\bar{\mathbb{T}} := (\bar{\Delta}, \bar{\mathfrak{A}}, \bar{\Phi})$.

$$\begin{aligned} \bar{\Delta} := \quad & \{(\infty, \mathit{spontaneously\_smokes}(jakob) \Rightarrow \mathit{smokes}(jakob)), \\ & \{(\infty, \mathit{spontaneously\_smokes}(kilian) \Rightarrow \mathit{smokes}(kilian)), \\ & (\ln(2), \mathit{smokes}(jakob) \Rightarrow \mathit{smokes}(jakob))\} \\ & (\ln(2), \mathit{smokes}(jakob) \Rightarrow \mathit{smokes}(kilian))\} \\ & (\ln(2), \mathit{smokes}(kilian) \Rightarrow \mathit{smokes}(jakob))\} \\ & (\ln(2), \mathit{smokes}(kilian) \Rightarrow \mathit{smokes}(kilian))\} \end{aligned}$$

$$\begin{aligned} \bar{\mathfrak{Ab}} := \quad & \{\mathit{spontaneously\_smokes}(jakob), \\ & \neg \mathit{spontaneously\_smokes}(jakob), \\ & \mathit{spontaneously\_smokes}(kilian), \\ & \neg \mathit{spontaneously\_smokes}(kilian), \\ & \neg \mathit{smokes}(jakob), \neg \mathit{smokes}(kilian)\} \end{aligned}$$

$$\begin{aligned} \bar{\Phi} := \quad & \{(\ln(0.2), \mathit{spontaneously\_smokes}(jakob)), \\ & (\ln(0.2), \mathit{spontaneously\_smokes}(kilian)), \\ & (\ln(0.8), \neg \mathit{spontaneously\_smokes}(jakob)), \\ & (\ln(0.8), \neg \mathit{spontaneously\_smokes}(kilian))\} \end{aligned}$$

A **weighted abductive logic program** is a tuple $\mathbb{P} := (\mathbf{P}, \mathfrak{Ab}, \Phi)$, consisting of an LP$^{\text{MLN}}$ program $\mathbf{P}$, a set of atoms $\mathfrak{Ab}$ and a Markov logic network $\Phi$. The **grounding** of $\mathbb{P}$ is then given by $\bar{\mathbb{P}} := (\bar{\mathbf{P}}, \bar{\mathfrak{Ab}}, \bar{\Phi})$ and we associate to $\mathbb{P}$ the distribution $\pi_{\mathbb{P}}(\_)$ that corresponds to its grounding $\bar{\mathbb{P}}$. Lastly, the **Bochman transformation** of a weighted abductive causal theory with default negation $\mathbb{T} := (\Delta, \mathfrak{Ab}, \Phi)$ is the weighted abductive logic program

$$\mathbb{P}(\mathbb{T}) := \big(\{(w, p \leftarrow \phi)\}_{(w,\phi\Rightarrow p)\in\Delta}, \{\text{atoms in } \mathfrak{Ab}\}, \Phi\big).$$

**Example 3.4.2.** The Bochman transformation of the weighted abductive causal theory $\mathbb{T}$ in Example 3.4.1 is the weighted abductive logic program $\mathbb{P}(\mathbb{T}) := (\mathbf{P}, \mathfrak{Ab}, \Phi)$ consisting of the LP$^{\text{MLN}}$ program

$$\begin{aligned} \mathbf{P}: \qquad & (\infty, \mathit{smokes}(X) \leftarrow \mathit{spontaneously\_smokes}(X)) \\ & (\ln(2), \mathit{smokes}(Y) \leftarrow \mathit{smokes}(X)) \end{aligned}$$

the abducibles $\mathfrak{Ab} := \{\mathit{spontaneously\_smokes}(X)\}$ and the Markov logic network

$$\Phi := \{(\ln(0.2), \mathit{spontaneously\_smokes}(X)), (\ln(0.8), \neg\mathit{spontaneously\_smokes}(X))\}.$$

As the semantics of both weighted abductive causal theories and weighted abductive logic programs is defined by grounding, we can conclude that Theorem 3.2.2 remains valid in the non-ground case. Weighted abductive logic programming thus emerges as a formalism specifically designed for causal reasoning. Hence, we define a causal interpretation of a framework in statistical relational artificial intelligence as follows.

**Definition 3.4.1** (Causal Interpretation)**.** A **causal interpretation** $\mathbb{C}(\_)$ of a framework that extends logic with probabilities is an embedding into weighted abductive logic programming.

Subsequently, we propose causal interpretations for the formalisms presented in Section 2.2.

### 3.4.1 A Causal Interpretation of Markov Logic Networks

We propose to interpret a Markov logic network $\Phi$ as the weighted abductive logic program

$$\mathbb{C}(\Phi) := (\emptyset, \{\text{atoms occurring in } \Phi\}, \Phi).$$

As $\Phi$ provides no causal explanations; all knowledge is a priori knowledge, not supporting queries about external interventions and counterfactuals.

### 3.4.2 A Causal Interpretation of LP$^{\text{MLN}}$ Programs

We interpret an LP$^{\text{MLN}}$ program $\mathbf{P}$ as the weighted abductive logic program $\mathbb{C}(\mathbf{P}) := (\mathbf{P}, \emptyset, \emptyset)$, i.e., the uncertainty captured in the soft weights $w \in \mathbb{R}$ is not explained. Suppose $\mathbf{P}$ is propositional. In that case, it allows us to answer queries about the effect of external interventions $\mathbf{i}$ by querying the **modified program $\mathbf{P}_\mathbf{i}$** in Definition 3.2.7. However, for counterfactual reasoning, we obtain the following result.

**Proposition 3.4.1.** *We obtain* $\pi_{\mathbb{C}(\boldsymbol{P})}(\phi|\boldsymbol{e}, \mathrm{do}(\boldsymbol{i})) = 0$*,* $\pi(\phi|\boldsymbol{e}, \mathrm{do}(\boldsymbol{i})) = 1$*, or* $\pi_{\mathbb{C}(\boldsymbol{P})}(\phi|\boldsymbol{e}, \mathrm{do}(\boldsymbol{i})) \in [0, 1]$ *for every counterfactual query* $\pi(\phi|\boldsymbol{e}, \mathrm{do}(\boldsymbol{i}))$*.*

*Proof.* As $\mathbb{C}(\mathbf{P})$ has no abducibles, i.e. $\mathfrak{A} = \emptyset$ we find that $\emptyset$ is the only $\mathfrak{A}$-structure with $\pi_\Phi(\emptyset) = 1$. Assuming that we find $0 < \pi_{\mathbf{P}_\mathbf{i}}(\phi) < 1$, we therefore obtain $l = 0$ as the lower bound and $u = 1$ as the upper bound of the counterfactual probability $\pi_{\mathbb{C}(\mathbf{P})}(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$. If $\pi_{\mathbf{P}_\mathbf{i}}(\phi) = 1$, we obtain $u = l = 1$. Finally, if $\pi_{\mathbf{P}_\mathbf{i}}(\phi) = 0$, we obtain $u = l = 0$. □

**Remark 3.4.1.** We conclude that LP$^{\text{MLN}}$ programs seem to be the right way to represent Boolean causal Bayesian networks (41, §1.3) in probabilistic logic programming.

### 3.4.3 A Causal Interpretation of ProbLog Programs

Assume **P** is a ProbLog program. In this case, we interpret **P** as the weighted abductive logic program $\mathbb{C}(\mathbb{P}) := (\mathbf{P}', \mathfrak{A}, \Phi)$, where $\mathbf{P}' := \{(\infty, C)\colon C \in \mathrm{LP}(\mathbf{P})\}$, $\mathfrak{A} := \{u(RF) : RF \in \mathrm{Facts}(\mathbf{P})\}$, and $\Phi := \{(\ln(\pi(RF)), u(RF)), (\ln(1 - \pi(RF)), \neg u(RF))\colon RF \in \mathrm{Facts}(\mathbf{P})\}$.

**Example 3.4.3.** Recall the ProbLog program **P** in Example 2.2.49.

**Random facts** Facts(**P**)**:**
$0.5 :: \mathit{spontanously\_smokes}(X)$

**Underlying Logic Program** LP(**P**) :
$\mathit{smokes}(X) \leftarrow \mathit{spontanously\_smokes}(X)$
$\mathit{smokes}(Y) \leftarrow \mathit{smokes}(X)$

The program **P** is interpreted as the weighted abductive logic program $\mathbb{C}(\mathbf{P}) := (\mathbf{P}', \mathfrak{A}, \Phi)$ consisting of the $\mathrm{LP}^{\mathrm{MLN}}$ program

$$\begin{aligned} \mathbf{P}' : \qquad & (\infty, \mathit{smokes}(X) \leftarrow \mathit{spontanously\_smokes}(X)) \\ & (\infty, \mathit{smokes}(Y) \leftarrow \mathit{smokes}(X)), \end{aligned}$$

the abducibles $\mathfrak{A} := \{\mathit{spontanously\_smokes}(X)\}$, and the a priori knowledge

$$\Phi := \quad \{(\ln(0.5), \mathit{spontanously\_smokes}(X)), \quad (\ln(0.5), \neg\mathit{spontanously\_smokes}(X))\}.$$

As desired, the causal interpretation preserves the semantics of the ProbLog program **P**.

**Proposition 3.4.2.** *For every structure $\omega$, we find $\pi_{\boldsymbol{P}}(\omega) = \pi_{\mathbb{C}(\boldsymbol{P})}(\omega)$.*

*Proof.* We may, without loss of generality, assume that **P** is a ProbLog program in a propositional alphabet $\mathfrak{P}$. Observe that a choice $\epsilon$ is consistent if and only if it corresponds to a consistent structure with respect to $\mathbb{C}(\mathbf{P})$, that is, $\mathrm{consistent}(\mathbf{P}) = \mathrm{consistent}(\mathbb{C}(\mathbf{P}))$.

Let $\epsilon \subseteq \mathfrak{A}$ be a consistent choice and let $\omega_1, \dots, \omega_n$ be the stable models of the logic program $\mathrm{LP}(\mathbf{P}) \cup \epsilon$. One verifies that $\omega_1, \dots, \omega_n$ are also the stable models $\omega$ with $\pi_{\mathbb{C}(\mathbf{P})}(\omega) \neq 0$ of the causal interpretation $\mathbb{C}(\mathbf{P})$ with explanation $\epsilon$ and that $\Phi$ induces the same distribution $\pi$ on the error terms in $\mathfrak{A}$ as the ProbLog program **P**. Hence, applying Parameterization 4 and the principle of indifference in Law 3, we can conclude the desired result:

$$\pi_{\mathbf{P}}(\omega_i) = \frac{1}{n} \cdot \pi(\epsilon | \, \mathrm{consistent}(\mathbf{P})) = \pi_{\mathbf{P}'}(\omega_i | \epsilon) \cdot \pi(\epsilon | \, \mathrm{consistent}(\mathbb{C}(\mathbf{P})) = \pi_{\mathbb{C}(\mathbf{P})}(\omega_i)$$

□

From now on, we assume that **P** is a ProbLog program in a propositional alphabet $\mathfrak{P}$. Our causal interpretation of the ProbLog program **P** gives rise to the following notion of external intervention.

**Definition 3.4.2** (Intervention in a ProbLog program)**.** Let **i** be a truth value assignment on the propositions $\mathbf{I} \subseteq \mathfrak{P}$. In this case, we model the external intervention of forcing **i** with the **modified ProbLog program $\mathbf{P_i}$**, which is defined by the modified logic program $\mathrm{LP}(\mathbf{P})_{\mathbf{i}}$ and the random facts Facts(**P**).

**Remark 3.4.2.** This is the same notion of intervention that Rückschloß and Weitkämper (49) introduce for acyclic ProbLog programs.

**Example 3.4.4.** Recall the ProbLog program $\mathbf{P}$ in Example 2.2.43.

> **Random facts** Facts($\mathbf{P}$)**:**
>
> $0.5 :: u_1$ $0.6 :: u_2$ $0.7 :: u_3$ $0.1 :: u_4$ $0.8 :: u_5$ $0.3 :: u_6$ $0.9 :: u_7$
>
> **Underlying Logic Program** LP($\mathbf{P}$) :
>
> $cloudy \leftarrow u_1$
>
> $rain \leftarrow cloudy, u_2$
>
> $sprinkler \leftarrow \neg cloudy, u_3$ $\qquad$ $sprinkler \leftarrow cloudy, u_4$
>
> $wet \leftarrow rain, u_5$ $\qquad$ $wet \leftarrow sprinkler, u_6$
>
> $slippery \leftarrow wet, u_7$

Assume that we manually turn the sprinkler off (on) in the situation of the program $\mathbf{P}$. This means that we intervene according to the truth value assignment $\mathbf{i} := \{\neg sprinkler\}$ ($\mathbf{i} := \{sprinkler\}$), resulting in the modified ProbLog program $\mathbf{P_i}$.

> **Random facts** Facts($\mathbf{P_i}$)**:**
>
> $0.5 :: u_1$ $0.6 :: u_2$ $0.7 :: u_3$ $0.1 :: u_4$ $0.8 :: u_5$ $0.3 :: u_6$ $0.9 :: u_7$
>
> **Underlying Logic Program** LP($\mathbf{P_i}$) :
>
> $cloudy \leftarrow u_1$
>
> $rain \leftarrow cloudy, u_2$
>
> $(sprinkler)$
>
> $wet \leftarrow rain, u_5$ $\qquad$ $wet \leftarrow sprinkler, u_6$
>
> $slippery \leftarrow wet, u_7.$

As intended, our concept of intervention, as defined in Definition 3.4.2, aligns with the causal interpretation $\mathbb{C}(\_)$.

**Proposition 3.4.3.** *For every truth value assignment $\boldsymbol{i}$ of propositions $\boldsymbol{I} \subseteq \mathfrak{P}$, we observe that the corresponding modified programs satisfy $\mathbb{C}(\boldsymbol{P_i}) = \mathbb{C}(\boldsymbol{P})_{\boldsymbol{i}}$.* □

Further, we obtain the following counterfactual reasoning.

**Definition 3.4.3** (Counterfactual Reasoning in ProbLog)**.** Let $\mathbf{e}, \mathbf{i}$ be truth value assignments on the subsets of propositions $\mathbf{E}, \mathbf{I} \subseteq \mathfrak{P}$, respectively. Further, let $\phi$ be a formula. In this case, we define

$$u := \sum_{\substack{\epsilon \text{ choice} \\ \pi_{\mathbf{P_i}}(\phi|\epsilon)>0}} \pi_{\mathbf{P}}(\epsilon|\mathbf{e}, \text{consistent}(\mathbf{P_i})) \qquad \text{and} \qquad l := \sum_{\substack{\epsilon \text{ choice} \\ \pi_{\mathbf{P_i}}(\phi|\epsilon)=1}} \pi_{\mathbf{P}}(\epsilon|\mathbf{e}, \text{consistent}(\mathbf{P_i})).$$

We then obtain for the probability that $\phi$ would be true if we had intervened according to $\mathbf{i}$ before observing the evidence $\mathbf{e}$ that $\pi_{\mathbf{P}}(\phi|\mathbf{e}, \text{do}(\mathbf{i})) \in [l, u]$.

**Example 3.4.5.** Assume that Kailin and Felix drink tea and the bell rings. If Kailin or Felix hears the bell, one of them stands up to open the door while the other remains seated. This can be modeled with the following ProbLog program $\mathbf{P}$.

> **Random facts** Facts($\mathbf{P}$)**:**
>
> $0.5 :: u_1$ $\qquad$ $0.5 :: u_2$
>
> **Underlying Logic Program** LP($\mathbf{P}$) :
>
> $felix \leftarrow \neg kailin, u_1$
>
> $kailin \leftarrow \neg felix, u_2$

As in Example 3.3.5, we then obtain $\pi_{\mathbf{P}}(felix|felix, \mathrm{do}(\emptyset)) \in [0.5, 1]$, which is consistent with Law 10. Further, assume that Felix answers the door; in this case, Kailin would have answered the door with probability $\pi_{\mathbf{P}}(kailin|felix, \mathrm{do}(\neg felix)) = 0.5$ if he had not opened the door.

As intended, the causal interpretation $\mathbb{C}(\_)$ is consistent with counterfactual reasoning.

**Proposition 3.4.4.** *For any truth value assignments $\boldsymbol{e}, \boldsymbol{i}$ on subsets of propositions $\boldsymbol{E}, \boldsymbol{I} \subseteq \mathfrak{P}$, we have $l_{\boldsymbol{P}} = l_{\mathbb{C}(\boldsymbol{P})}$ and $u_{\boldsymbol{P}} = u_{\mathbb{C}(\boldsymbol{P})}$, where $\pi_{\boldsymbol{P}}(\phi|\boldsymbol{e}, \mathrm{do}(\boldsymbol{i})) \in [l_{\boldsymbol{P}}, u_{\boldsymbol{P}}]$ and $\pi_{\mathbb{C}(\boldsymbol{P})}(\phi|\boldsymbol{e}, \mathrm{do}(\boldsymbol{i})) \in [l_{\mathbb{C}(\boldsymbol{P})}, u_{\mathbb{C}(\boldsymbol{P})}]$, as discussed in Section 3.3 and defined in Definition 3.4.3.* □

We are now interested in ProbLog programs $\mathbf{P}$ that assign a precise value to each counterfactual probability $\pi(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$.

**Definition 3.4.4** (ProbLog Programs with Full Explanation)**.** A ProbLog program $\mathbf{P}$ offers **full explanation** if, for every choice $\epsilon$ with $\pi(\epsilon) > 0$ and for every assignment of truth values $\mathbf{i}$ in a set of propositions $\mathbf{I} \subseteq \mathfrak{P}$, the logic program $\mathrm{LP}(\mathbf{P})_{\mathbf{i}} \cup \epsilon$ has at most one stable model $\omega$.

**Proposition 3.4.5.** *Every ProbLog program $\boldsymbol{P}$ with a stratified logic program $\mathrm{LP}(\boldsymbol{P})$ provides full explanations.*

*Proof.* The desired result follows from Theorem 2.2.2 and the observation that for every truth value assignment $\mathbf{i}$ on a set of propositions $\mathbf{I} \subseteq \mathfrak{P}$ the program $\mathrm{LP}(\mathbf{P})_{\mathbf{i}}$ is again stratified. □

**Example 3.4.6.** The acyclic ProbLog programs in Example 3.4.4 provide full explanation, whereas the program in Example 3.4.5 does not provide full explanations.

Now, ProbLog programs providing full explanation are those that answer each counterfactual query with a sharp probability.

**Theorem 3.4.6.** *The ProbLog program $\boldsymbol{P}$ answers every counterfactual query $\pi(\phi|\boldsymbol{e}, \mathrm{do}(\boldsymbol{i}))$ with a sharp probability if and only if it provides full explanation. In other words, we find*

$$\pi(\phi|\boldsymbol{e}, \mathrm{do}(\boldsymbol{i})) := l = u$$

*for all estimates in Definition 3.4.3 if and only if the program $\boldsymbol{P}$ provides full explanation.*

*Proof.* As $\mathbf{P}$ provides full explanation, for every consistent choice $\epsilon$ with $\pi(\epsilon) > 0$ there exists a unique stable model $\omega$ of the modified logic program $\mathrm{LP}(\mathbf{P})_{\mathbf{i}} \cup \epsilon$. Hence, we find

$$\pi_{\mathbf{P}_{\mathbf{i}}}(\phi|\epsilon) = \begin{cases} 1, & \text{if } \omega \models \phi \\ 0, & \text{else} \end{cases}$$

and we conclude

$$l = \sum_{\substack{\epsilon \text{ consistent choice} \\ \omega \text{ stable model of } \mathrm{LP}(\mathbf{P})_{\mathbf{i}} \cup \epsilon \\ \omega \models \phi}} \pi_{\mathbf{P}}(\epsilon|\mathbf{e}, \mathrm{consistent}(\mathbf{P}_{\mathbf{i}})) = u.$$

Further, assume the ProbLog program $\mathbf{P}$ does not provide full explanation, i.e. there is a truth value assignment $\mathbf{i}$ and choice $\epsilon$ with $\pi(\epsilon) > 0$ such that the modified logic program $\mathrm{LP}(\mathbf{P})_{\mathbf{i}} \cup \epsilon$ has two stable models $\omega_{1/2}$, i.e. $0 < \pi_{\mathbf{P}_{\mathbf{i}}}(\omega_1|\epsilon) < 1$. We find for the estimates $l \leq u$ of $\pi(\omega_1|\omega_2, \mathrm{do}(\mathbf{i}))$ that $l = 0$ and $u = \pi_{\mathbf{P}}(\epsilon|\omega_1) = 1$. □

Theorem 3.4.6 states that within ProbLog, programs providing full explanation exactly represent the probabilistic Aristotelian knowledge necessary for well-defined counterfactual reasoning, i.e., reasoning that yields a unique probability for every counterfactual query. For ProbLog programs that offer full explanations, we can further adapt the twin network method proposed by Balke and Pearl (5) to compute these counterfactual probabilities. Let us start by defining twin programs.

**Definition 3.4.5** (Twin Program)**.** Create two copies, $\mathfrak{P}^r$ and $\mathfrak{P}^a$, of our propositional alphabet $\mathfrak{P}$ – one to represent the real world and the other to represent the alternative world. Further, set $u(RF)^a := u(RF)^r := u(RF)$ for all $RF \in \text{Facts}(\mathbf{P})$, establishing mappings $\_^{a/r}$ for formulas, truth value assignments, programs, etc. The **twin program** $\mathbf{P}^T$ of $\mathbf{P}$ is then the ProbLog program defined by $\text{LP}\left(\mathbf{P}^T\right) := \text{LP}(\mathbf{P})^a \cup \text{LP}(\mathbf{P})^r$ and $\text{Facts}\left(\mathbf{P}^T\right) := \text{Facts}(\mathbf{P})$.

**Example 3.4.7.** Recall the ProbLog program $\mathbf{P}$ in Example 2.2.43.

**Random facts** Facts($\mathbf{P}$)**:**

$0.5 :: u_1$ $0.6 :: u_2$ $0.7 :: u_3$ $0.1 :: u_4$ $0.8 :: u_5$ $0.3 :: u_6$ $0.9 :: u_7$

**Underlying Logic Program** LP($\mathbf{P}$) :

$\mathit{cloudy} \leftarrow u_1$

$\mathit{rain} \leftarrow \mathit{cloudy}, u_2$

$\mathit{sprinkler} \leftarrow \neg \mathit{cloudy}, u_3$ $\qquad$ $\mathit{sprinkler} \leftarrow \mathit{cloudy}, u_4$

$\mathit{wet} \leftarrow \mathit{rain}, u_5$ $\qquad$ $\mathit{wet} \leftarrow \mathit{sprinkler}, u_6$

$\mathit{slippery} \leftarrow \mathit{wet}, u_7$

We obtain the following twin program $\mathbf{P}^T$ of the program $\mathbf{P}$.

**Random facts** Facts($\mathbf{P}^T$)**:**

$0.5 :: u_1$ $0.6 :: u_2$ $0.7 :: u_3$ $0.1 :: u_4$ $0.8 :: u_5$ $0.3 :: u_6$ $0.9 :: u_7$

**Underlying Logic Program** LP($\mathbf{P}^T$) :

$\mathit{cloudy}^r \leftarrow u_1$ $\qquad$ $\mathit{cloudy}^a \leftarrow u_1$

$\mathit{rain}^r \leftarrow \mathit{cloudy}^r, u_2$ $\qquad$ $\mathit{rain}^a \leftarrow \mathit{cloudy}^a, u_2$

$\mathit{sprinkler}^r \leftarrow \neg \mathit{cloudy}^r, u_3$ $\qquad$ $\mathit{sprinkler}^a \leftarrow \neg \mathit{cloudy}^a, u_3$

$\mathit{sprinkler}^r \leftarrow \mathit{cloudy}^r, u_4$ $\qquad$ $\mathit{sprinkler}^a \leftarrow \mathit{cloudy}^a, u_4$

$\mathit{wet}^r \leftarrow \mathit{rain}^r, u_5$ $\qquad$ $\mathit{wet}^a \leftarrow \mathit{rain}^a, u_5$

$\mathit{wet}^r \leftarrow \mathit{sprinkler}^r, u_6$ $\qquad$ $\mathit{wet}^a \leftarrow \mathit{sprinkler}^a, u_6$

$\mathit{slippery}^r \leftarrow \mathit{wet}^r, u_7$ $\qquad$ $\mathit{slippery}^a \leftarrow \mathit{wet}^a, u_7.$

Twin programs allow us to compute counterfactual probabilities.

**Lemma 3.4.7.** *Let $\boldsymbol{P}$ be a ProbLog program that offers full explanation. For every choice $\epsilon$ and every truth value assignment $\boldsymbol{i}$ on a set of propositionts $\boldsymbol{I} \subseteq \mathfrak{P}$ we find*

$$\pi_{\boldsymbol{P}}(\epsilon |\, \text{consistent}(\boldsymbol{P_i})) = \pi_{\boldsymbol{P}^T_{i^a}}(\epsilon).$$

*Proof.* Denote by $\pi$ the distribution on the choices of $\mathbf{P}$, $\mathbf{P_i}$ and $\mathbf{P}^T_{\mathbf{i}^a}$, respectively, that is induced by the random facts in $\text{Facts}(\mathbf{P}) = \text{Facts}(\mathbf{P_i}) = \text{Facts}(\mathbf{P}^T_{\mathbf{i}^a})$.

Let $\epsilon$ be a choice of $\mathbf{P}$ that is consistent with respect to both programs $\mathbf{P}$ and $\mathbf{P_i}$. In this case, we obtain unique stable models $\omega_\mathbf{i}$ of $\text{LP}(\mathbf{P})_\mathbf{i} \cup \epsilon$ and $\omega$ of $\text{LP}(\mathbf{P}) \cup \epsilon$. We observe that $\omega^r \cup \omega^a_\mathbf{i}$ yields a stable model of $\mathbf{P}^T_{\mathbf{i}^a}$, that is, $\epsilon$ is consistent with respect to $\mathbf{P}^T_{\mathbf{i}^a}$.

If $\epsilon$ is a consistent choice of $\mathbf{P}^T_{\mathbf{i}^a}$, we obtain a unique stable model $\omega$ of $\text{LP}(\mathbf{P}^T_{\mathbf{i}^a}) \cup \epsilon$. Now we observe that $\omega \cap \mathfrak{P}^a$ yields a stable model of $\text{LP}(\mathbf{P_i}) \cup \epsilon$ and $\omega \cap \mathfrak{P}^r$ yields a stable model of $\text{LP}(\mathbf{P}) \cup \epsilon$. Hence, $\epsilon$ is consistent with respect to both the programs $\mathbf{P}$ and $\mathbf{P_i}$.

In summary, we find $\text{consistent}(\mathbf{P}^T_{\mathbf{i}^a}) = \text{consistent}(\mathbf{P}) \cap \text{consistent}(\mathbf{P_i})$ and

$$\pi_\mathbf{P}(\epsilon |\, \text{consistent}(\mathbf{P_i})) = \pi(\epsilon |\, \text{consistent}(\mathbf{P_i}), \text{consistent}(\mathbf{P_i})) = \pi(\epsilon |\, \text{consistent}(\mathbf{P}^T_{\mathbf{i}^a})) = \pi_{\mathbf{P}^T_{\mathbf{i}^a})}(\epsilon)$$

for all choices $\epsilon$ of $\mathbf{P}$. □

**Theorem 3.4.8.** *If the ProbLog program $\boldsymbol{P}$ provides full explanation, we have*

$$\pi_{\boldsymbol{P}}(\phi|\boldsymbol{e}, \mathrm{do}(\boldsymbol{i})) = \pi_{\boldsymbol{P}^T_{\boldsymbol{i}^a}}(\phi^a|\boldsymbol{e}^r)$$

*for every formula $\phi$ and every truth value assignments $\boldsymbol{e}, \boldsymbol{i}$ on sets of propositions $\boldsymbol{E}, \boldsymbol{I} \subseteq \mathfrak{P}$.*

*Proof.* Let us calculate.

$$\pi_{\mathbf{P}}(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i})) = \sum_{\substack{\epsilon \text{ consistent choice of } \mathbf{P} \text{ and } \mathbf{P_i} \\ \omega \text{ stable model of } \mathrm{LP}(\mathbf{P})_{\mathbf{i}} \cup \epsilon \\ \omega \models \phi}} \pi_{\mathbf{P}}(\epsilon|\mathbf{e}, \mathrm{consistent}(\mathbf{P_i})) =$$

$$= \sum_{\substack{\epsilon \text{ consistent choice of } \mathbf{P} \text{ and } \mathbf{P_i} \\ \omega_{\mathbf{i}} \text{ stable model of } \mathrm{LP}(\mathbf{P})_{\mathbf{i}} \cup \epsilon \\ \omega \text{ stable model of } \mathrm{LP}(\mathbf{P}) \cup \epsilon \\ \omega \models \mathbf{e},\ \omega_{\mathbf{i}} \models \phi}} \pi_{\mathbf{P}}(\epsilon|\mathbf{e}, \mathrm{consistent}(\mathbf{P_i})) =$$

$$= \frac{1}{\pi_{\mathbf{P}}(\mathbf{e}|\,\mathrm{consistent}(\mathbf{P_i}))} \sum_{\substack{\epsilon \text{ consistent choice of } \mathbf{P} \text{ and } \mathbf{P_i} \\ \omega_{\mathbf{i}} \text{ stable model of } \mathrm{LP}(\mathbf{P})_{\mathbf{i}} \cup \epsilon \\ \omega \text{ stable model of } \mathrm{LP}(\mathbf{P}) \cup \epsilon \\ \omega \models \mathbf{e},\ \omega_{\mathbf{i}} \models \phi}} \pi_{\mathbf{P}}(\epsilon, \mathbf{e}|\,\mathrm{consistent}(\mathbf{P_i})) =$$

$$= \frac{1}{\pi_{\mathbf{P}}(\mathbf{e}|\,\mathrm{consistent}(\mathbf{P_i}))} \sum_{\substack{\epsilon \text{ consistent choice of } \mathbf{P} \text{ and } \mathbf{P_i} \\ \omega_{\mathbf{i}} \text{ stable model of } \mathrm{LP}(\mathbf{P})_{\mathbf{i}} \cup \epsilon \\ \omega \text{ stable model of } \mathrm{LP}(\mathbf{P}) \cup \epsilon \\ \omega \models \mathbf{e},\ \omega_{\mathbf{i}} \models \phi}} \pi_{\mathbf{P}}(\epsilon|\,\mathrm{consistent}(\mathbf{P_i})) \overset{\text{Lemma 3.4.7}}{=}$$

$$= \frac{1}{\pi_{\mathbf{P}}(\mathbf{e}|\,\mathrm{consistent}(\mathbf{P_i}))} \sum_{\substack{\epsilon \text{ consistent choice} \\ \omega \text{ stable model of } \mathrm{LP}(\mathbf{P}^T)_{\mathbf{i}^a} \cup \epsilon \\ \omega \models \mathbf{e}^r,\ \omega_{\mathbf{i}} \models \phi^a}} \pi_{\mathbf{P}^T_{\mathbf{i}^a}}(\epsilon) =$$

$$= \pi_{\mathbf{P}^T_{\mathbf{i}^a}}(\phi^a, \mathbf{e}^r) \cdot \pi_{\mathbf{P}}(\mathbf{e}, \mathrm{consistent}(\mathbf{P_i}))^{-1} =$$

$$= \pi_{\mathbf{P}^T_{\mathbf{i}^a}}(\phi^a, \mathbf{e}^r) \left( \sum_{\substack{\epsilon \text{ consistent choice} \\ \omega \text{ stable model of } \mathrm{LP}(\mathbf{P}) \cup \epsilon \\ \omega \models \mathbf{e}}} \pi_{\mathbf{P}}(\epsilon|\,\mathrm{consistent}(\mathbf{P_i})) \right)^{-1} \overset{\text{Lemma 3.4.7}}{=}$$

$$= \pi_{\mathbf{P}^T_{\mathbf{i}^a}}(\phi^a, \mathbf{e}^r) \left( \sum_{\substack{\epsilon \text{ consistent choice} \\ \omega \text{ stable model of } \mathrm{LP}(\mathbf{P}^T)_{\mathbf{i}^a} \cup \epsilon \\ \omega \models \mathbf{e}^r}} \pi_{\mathbf{P}^T_{\mathbf{i}^a}}(\epsilon) \right)^{-1} = \frac{\pi_{\mathbf{P}^T_{\mathbf{i}^a}}(\phi^a, \mathbf{e}^r)}{\pi_{\mathbf{P}^T_{\mathbf{i}^a}}(\mathbf{e}^r)} = \pi_{\mathbf{P}^T_{\mathbf{i}^a}}(\phi^a|\mathbf{e}^r)$$

□

Kiesel et al. (34) employ Theorem 3.4.8 to compute counterfactual probabilities for acyclic ProbLog programs. Notably, Kiesel's WhatIf solver extends his aspmc solver, utilizing Theorem 3.4.8 to calculate counterfactual probabilities.

**Example 3.4.8.** In Example 3.4.7, we may observe the evidence $\mathbf{e} := \{\mathit{sprinkler}, \mathit{slippery}\}$ that the sprinkler is on and that the road is slippery. Assume now that we are interested in the counterfactual probability $\pi(\mathit{slippery}|\mathit{sprinkler}, \mathit{slippery}, \mathrm{do}(\neg \mathit{sprinkler}))$ for the road to be slippery if the sprinkler were off. In this case, we query the modified twin program $\mathbf{P}^T_{\neg \mathit{sprinkler}^a}$

**Random facts** $\mathrm{Facts}(\mathbf{P}^T_{\neg \mathit{sprinkler}^a})$**:**

$0.5 :: u_1$ $0.6 :: u_2$ $0.8 :: u_3$ $0.1 :: u_4$ $0.7 :: u_5$ $0.3 :: u_6$ $0.9 :: u_7$

**Underlying Logic Program** $\mathrm{LP}(\mathbf{P}^T_{\neg \mathit{sprinkler}^a})$ :

$$\begin{array}{ll}
\mathit{cloudy}^r \leftarrow u_1 & \mathit{cloudy}^a \leftarrow u_1 \\
\mathit{rain}^r \leftarrow \mathit{cloudy}^r, u_2 & \mathit{rain}^a \leftarrow \mathit{cloudy}^a, u_2 \\
\mathit{sprinkler}^r \leftarrow \neg \mathit{cloudy}^r, u_3 & \\
\mathit{sprinkler}^r \leftarrow \mathit{cloudy}^r, u_4 & \\
\mathit{wet}^r \leftarrow \mathit{rain}^r, u_5 & \mathit{wet}^a \leftarrow \mathit{rain}^a, u_5 \\
\mathit{wet}^r \leftarrow \mathit{sprinkler}^r, u_6 & \mathit{wet}^a \leftarrow \mathit{sprinkler}^a, u_6 \\
\mathit{slippery}^r \leftarrow \mathit{wet}^r, u_7 & \mathit{slippery}^a \leftarrow \mathit{wet}^a, u_7.
\end{array}$$

for the probability

$$\pi(\mathit{slippery}|\mathit{sprinkler}, \mathit{slippery}, \mathrm{do}(\neg \mathit{sprinkler})) = \pi(\mathit{slippery}^a|\mathit{sprinkler}^r, \mathit{slippery}^r) = 0.140...$$

Lastly, we describe the fragment of weighted abductive logic programming that corresponds to ProbLog. To this aim, we recall Reichenbach's common cause principle: "If an improbable coincidence has occurred, there must exist a common cause." (45, Ch. 19) In other words, two random variables are dependent either because of a causal relationship or because they share a common cause. Here, we weaken this principle and assert that all dependencies stem either from a causal relationship or from a common cause or from deriving contradictions with causal reasoning. We say that a weighted abductive logic program explains all dependencies if there is no uncertainty in its clauses and if the prior knowledge $\Phi$ interprets the abducibles as mutually independent random variables.

**Definition 3.4.6** (Explaining Dependencies)**.** A weighted abductive logic program $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ **explains all dependencies** if for every weighted clause $(w, C) \in \mathbf{P}$ we find $w = \infty$ and if $\Phi$ interprets the abducibles $\mathfrak{A}$ as mutually independent Boolean random variables.

We now obtained that every weighted abductive logic program that explains all dependencies corresponds to a ProbLog program and vice versa.

**Proposition 3.4.9.** *Let $\mathbb{P} := (\boldsymbol{P}, \mathfrak{A}, \Phi)$ be a weighted abductive logic program that explains all dependencies. In this case, there exists a ProbLog program $\boldsymbol{P}$ such that the causal interpretation $\mathbb{C}(\boldsymbol{P})$ yields the same answer to every query as $\mathbb{P}$ and vice versa.*

*Proof.* Since the abducibles $\mathfrak{A}$ are interpreted as mutually independent random variables, we may, without loss of generality, assume that $\Phi := \{(\ln(\pi(u)), u), (\ln(1 - \pi(u)), \neg u) : u \in \mathfrak{A}\}$. Now let $\mathbf{P}'$ be the ProbLog program that is given by the logic program $\mathrm{LP}(\mathbf{P}') := \{C : (\infty, C) \in \mathbf{P}\}$ and by $\mathrm{Facts}(\mathbf{P}') := \{\pi(u) :: u : u \in \mathfrak{A}\}$. We then find $\mathbb{C}(\mathbf{P}) = \mathbb{P}$ as desired. For the other direction, we observe that the causal interpretation $\mathbb{C}(\mathbf{P})$ of each ProbLog program $\mathbf{P}$ explains all dependencies. □

**Remark 3.4.3.** Similarly, ProbLog programs correspond to the Bochman transformations of Boolean causal models that interpret the external variables as mutually independent error terms.

In conclusion, our discussion in this section and Section 3.3 leads to the following understanding: Representing (probabilistic) Aristotelian knowledge by weighted abductive logic programs, ProbLog programs, which offer full explanation, precisely capture Aristotelian knowledge that is necessary for well-defined counterfactual reasoning in the sense of Pearl (41), i.e., reasoning that yields a unique probability for every counterfactual query.

### 3.4.4 A Causal Interpretation of Logic Programs with Annotated Disjunctions

Here, we aim at a causal interpretation of LPADs (62). We focus on propositional LPADs, as the semantics of non-ground LPADs is defined by grounding. Therefore, our results easily extend to the non-ground case. Fix a propositional alphabet $\mathfrak{P}$ and assume $\mathbf{P}$ is an LPAD. We interpret $\mathbf{P}$ as the weighted abductive logic program

$$\mathbb{C}(\mathbf{P}) := (\mathbf{P}', \mathfrak{A}, \Phi),$$

where $\mathbf{P}' := \{(\infty, h_{\sigma(C)}(C) \leftarrow \text{body}(C) \cup \{\sigma\}) \colon C \in \mathbf{P}, \sigma \text{ selection}, \sigma(C) \neq \perp\}$, $\mathfrak{A}$ is the set of selections, and $\Phi$ consists of the weak constraints $(\ln(\pi(\sigma)), \sigma)$ for the selections $\sigma$ along with hard constraints indicating mutual exclusivity among selections. In this context, we observe that $\mathbb{C}(\_)$ preserves the semantics of $\mathbf{P}$, that is, $\pi_{\mathbf{P}}(\phi) = \pi_{\mathbb{C}(\mathbf{P})}(\phi)$ for all formulas $\phi$. In the propositional case, we obtain the notion of intervention discussed in Section 2.3.2 and the counterfactual reasoning introduced in Procedure 2.3.1. Furthermore, we note that the LPAD and ProbLog transformations, as defined in Definition 2.2.31 and 2.2.32, respectively, preserve the outcome of counterfactual queries.

**Lemma 3.4.10.** *Choose a literal $l \in \{p, \neg p\}$ for a proposition $p \in \mathfrak{P}$.*

*i) In the situation of Theorem 2.2.5, for every choice $\epsilon \in \epsilon(\sigma)$ the logic program $\boldsymbol{P}_l^\sigma$ and the logic program $\text{LP}(\text{Prob}(\boldsymbol{P}))_l \cup \epsilon$ yield the same stable models.*

*ii) In the situation of Theorem 2.2.6, for every selection $\sigma$ of $\text{LPAD}(\boldsymbol{P})$, the program $\text{LPAD}(\boldsymbol{P})_l^\sigma$ and the program $\text{LP}(\boldsymbol{P})_l \cup \epsilon(\sigma)$ yield the same stable models.*

*Proof.* We provide only a proof for i) since ii) can be proven analogously. From Theorem 2.2.5, we deduce that the programs $\mathbf{P}^\sigma$ and $\text{LP}(\text{Prob}(\mathbf{P})) \cup \epsilon$ share the same stable model $\omega$. Due to the modular nature of logic programs, this behavior remains unchanged if we eliminate all clauses with $p$ in the head of both programs. Finally, the desired behavior is not disturbed by adding the fact $p$ to both programs. □

**Theorem 3.4.11** (Consistency with ProbLog – Part 1)**.** *Let $\boldsymbol{I}$ and $\boldsymbol{E}$ be subsets of propositions in $\mathfrak{P}$, each associated with the assignment of truth values $\boldsymbol{i}$ and $\boldsymbol{e}$, respectively. Additionally, fix a formula $\phi$ and assume that the logic programs $\boldsymbol{P}^\sigma$ and $\boldsymbol{P}_{\boldsymbol{i}}^\sigma$ produce unique stable models $\omega(\sigma)$ and $\omega_{\boldsymbol{i}}(\sigma)$, respectively, for all selections $\sigma$ of $\boldsymbol{P}$. In this context, we have*

$$\pi_{\boldsymbol{P}}(\phi|\boldsymbol{e}, \text{do}(\boldsymbol{i})) = \pi_{\text{Prob}(\boldsymbol{P})}(\phi|\boldsymbol{e}, \text{do}(\boldsymbol{i})).$$

*Proof.* By Theorem 2.2.5 and Lemma 3.4.10, the right-hand side of (2.11) for $\mathbf{P}$ is the sum of conditional probabilities $\pi_{\text{Prob}(\mathbf{P})}(\epsilon|\mathbf{e})$ for all choices $\epsilon$ of $\text{Prob}(\mathbf{P})$ such that $\omega \models \mathbf{e}$ for the stable model $\omega$ of $\text{LP}(\text{Prob}(\mathbf{P})) \cup \epsilon$, and $\omega_{\mathbf{i}} \models \phi$ for the stable model $\omega_{\mathbf{i}}$ of $\text{LP}(\text{Prob}(\mathbf{P}))_{\mathbf{i}} \cup \epsilon$. These choices correspond precisely to scenarios where the query $\phi$ is true after the intervention, while the observation $\mathbf{e}$ is true before the intervention. Hence, Definition 3.4.3 computes the same value as Procedure 2.3.1. □

**Theorem 3.4.12** (Consistency with CP-Logic – Part 2)**.** *Revisiting the scenario described in Theorem 3.4.11, and assuming that $\boldsymbol{P}$ is a ProbLog program offering full explanation, we obtain the equality*

$$\pi_{\text{LPAD}(\boldsymbol{P})}(\phi|\boldsymbol{e}, \text{do}(\boldsymbol{i})) = \pi_{\boldsymbol{P}}(\phi|\boldsymbol{e}, \text{do}(\boldsymbol{i})).$$

*Proof.* Using Theorem 2.2.6 instead of Theorem 2.2.5, we can establish an analogous proof to that of Theorem 3.4.11. □

However, intervening in LPADs could yield counterintuitive results.

**Example 3.4.9** (Rückschloß and Weitkämper (50))**.** Assume that we are throwing a die. We can throw one, two, three, four, five, or six, each with a probability of 1/6. This scenario can be modeled with the LPAD clause:

$$one : \frac{1}{6}; two : \frac{1}{6}; three : \frac{1}{6}; four : \frac{1}{6}; five : \frac{1}{6}; six : \frac{1}{6}$$

Now, if we intervene by simply turning over the die to show the number four, our program is modified accordingly:

$$\begin{aligned} & four : 1 \\ & one : \frac{1}{6}; two : \frac{1}{6}; three : \frac{1}{6}; five : \frac{1}{6}; six : \frac{1}{6} \end{aligned}$$

In particular, we obtain a probability of 1/6 for the die to show both one and four at the same time, contradicting our daily experience.

Given that counterfactual reasoning in LPADs can be effectively modeled in ProbLog and avoiding unrealistic results as in Example 3.4.9, our forthcoming investigation of counterfactual reasoning will utilize the ProbLog language instead of LPAD.

## 3.5 First Result: A Unifying Framework for Causal Knowledge

We observe that weighted abductive logic programming is an effective tool for comparing the causal reasoning capabilities across various frameworks within statistical relational artificial intelligence. Notably, ProbLog programs that provide full explanation emerge as the preferred formalism for addressing Boolean counterfactual reasoning. In the next chapter, we delve into a detailed exploration of the relationship between counterfactual reasoning and acyclic ProbLog programs. Working out the causal query types, i.e. queries for the effect of external interventions and counterfactual queries, for the non-ground case would be an interesting direction for future work.

# Chapter 4

# Extracting Causal Knowledge from Counterfactuals

## Statement of Contribution

This chapter builds upon the material presented by Rückschloß and Weitkämper (51). Here, Kilian Rückschloß contributed the main idea and proofs of all results, while Felix Weitkämper contributed the motivation and an example of two programs that give rise to the same counterfactual reasoning (51, Example 14).

This chapter improves on that contribution with:

- additional examples;
- more detailed proofs, explanations, and correction of minor mistakes;
- an in-depth discussion of the assumptions underlying the results.

In Section 3.3, we introduce counterfactual reasoning based on probabilistic Aristotelian knowledge, represented by weighted abductive logic programs $\mathbb{P} := (\mathbf{P}, \mathfrak{A}, \Phi)$ in a propositional alphabet $\mathfrak{P}$. Let $\phi$ be a formula, and let $\mathbf{e}, \mathbf{i}$ be truth value assignments on the propositions in $\mathbf{E}, \mathbf{I} \subseteq \mathfrak{P} \setminus \mathfrak{A}$, respectively. Generally, the counterfactual probability $\pi(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$ – indicating the likelihood of $\phi$ holding if we had enforced $\mathbf{i}$ before observing the evidence $\mathbf{e}$ – is underdetermined by the program $\mathbb{P}$ and can only be estimated within an interval $\pi(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i})) \in [l, u]$.

However, in Section 3.4.3, we demonstrate that the probabilistic Aristotelian knowledge required to compute sharp counterfactual probabilities $\pi(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$ is precisely provided by ProbLog programs $\mathbf{P}$ that offer full explanation. In this chapter, we now investigate the relationship between counterfactual reasoning and probabilistic Aristotelian knowledge, as stated by these ProbLog programs. Specifically, we study the information about a ProbLog program $\mathbf{P}$ provided by the counterfactual probabilities $\pi(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$ and ask for the notion of program equivalence induced by counterfactual reasoning.

Recall the ProbLog program $\mathbf{P}$ in Example 2.2.43.

**Random facts** Facts($\mathbf{P}$)**:**

$0.5 :: u_1$ $0.6 :: u_2$ $0.7 :: u_3$ $0.1 :: u_4$ $0.8 :: u_5$ $0.3 :: u_6$ $0.9 :: u_7$

**Underlying Logic Program** LP($\mathbf{P}$) :

$\mathit{cloudy} \leftarrow u_1$

$\mathit{rain} \leftarrow \mathit{cloudy}, u_2$

$\mathit{sprinkler} \leftarrow \neg \mathit{cloudy}, u_3$ $\qquad$ $\mathit{sprinkler} \leftarrow \mathit{cloudy}, u_4$

$\mathit{wet} \leftarrow \mathit{rain}, u_5$ $\qquad$ $\mathit{wet} \leftarrow \mathit{sprinkler}, u_6$

$\mathit{slippery} \leftarrow \mathit{wet}, u_7$

Hence, we assume that it is cloudy, denoted *cloudy*, with probability 0.5. If cloudy, it rains, denoted *rain*, with probability 0.6. The sprinkler is then switched on, denoted *sprinkler*, by a weather sensor with a probability of 0.1 if it is cloudy and a probability of 0.7 if it is sunny. The road pavement is wet, denoted *wet*, with a probability of 0.8 if it rains and with a probability of 0.3 if the sprinkler is on. Finally, a wet road is slippery, denoted *slippery*, with a probability of 0.9.

Based on the program $\mathbf{P}$, we can compute counterfactual probabilities, such as

$$\pi(\mathit{slippery} \mid \mathit{sprinkler}, \mathit{slippery}, \mathrm{do}(\neg \mathit{sprinkler})) := \pi(\mathit{slippery}^a \mid \mathit{sprinkler}^r, \mathit{slippery}^r) = 0.140...,$$

indicating a likelihood of 0.140... for the road to be slippery if we had switched the sprinkler off before observing that the road is slippery while the sprinkler is indeed on. As the program $\mathbf{P}$ is acyclic, Theorem 3.4.6 guarantees that $\mathbf{P}$ provides a concrete value for every counterfactual probability $\pi(\phi|\mathbf{e}, \mathrm{do}(\mathbf{i}))$.

At this point, we address the following questions. What information do counterfactual probabilities, such as $\pi(\mathit{slippery}|\mathit{sprinkler}, \mathit{slippery}, \mathrm{do}(\neg \mathit{sprinkler}))$, capture about the program $\mathbf{P}$? What are the ProbLog programs $\mathbf{P}'$ in the alphabet $\mathfrak{P}$ that produce the same counterfactual probabilities as $\mathbf{P}$?

Considering the program $\mathbf{P}$, we recognize that every clause in the underlying logic program LP($\mathbf{P}$) mentions a unique error term $u_1, ..., u_6$. We conclude that the error terms serve as switches for clauses in a logic program. Suppressing the error terms, we therefore identify $\mathbf{P}$ with the following ProbLog clauses, indicating that we choose these clauses independently with the corresponding probability.

$0.5 :: \mathit{cloudy}$

$0.6 :: \mathit{rain} \leftarrow \mathit{cloudy}$

$0.7 :: \mathit{sprinkler} \leftarrow \neg \mathit{cloudy}$ $\qquad$ $0.1 :: \mathit{sprinkler} \leftarrow \mathit{cloudy}$

$0.8 :: \mathit{wet} \leftarrow \mathit{rain}$ $\qquad$ $0.3 :: \mathit{wet} \leftarrow \mathit{sprinkler}$

$0.9 :: \mathit{slippery} \leftarrow \mathit{wet}$

Recall from Section 2.2.4 that a ProbLog program $\mathbf{P}$ lies in the **fragment of ProbLog clauses** if each clause in the underlying logic program $\mathrm{LP}(\mathbf{P})$ mentions a unique error term. In this case, the ProbLog program $\mathbf{P}$ can be represented by ProbLog clauses. This viewpoint contrasts with the usual presentation of ProbLog, where one would introduce an extra error term for every ProbLog clause. From this perspective, ProbLog clauses are just a syntactic sugar to conveniently represent general ProbLog programs. In the context of causal reasoning, however, the error terms have a semantic meaning, i.e., they capture the influence of hidden variables not explicitly modeled in our program. Adopting this viewpoint, the fragment of ProbLog clauses should indeed be considered as a fragment of general ProbLog programs.

In this chapter, we restrict ourselves to studying acyclic ProbLog programs within the fragment of ProbLog clauses. Section 4.1 then presents examples of such ProbLog programs that yield the same counterfactual probabilities and introduces the notion of a proper ProbLog program in normal form that satisfies Occam's razor. Furthermore, in Section 4.2, we prove our main result, Theorem 4.2.6, stating that counterfactual reasoning determines acyclic proper ProbLog programs in normal form that satisfy Occam's razor up to syntactic equality. Hence, we obtain a fragment of probabilistic Aristotelian knowledge uniquely determined by counterfactual reasoning.

## 4.1 Knowledge Underdetermined by Counterfactual Reasoning

For the remainder of this chapter, we fix a propositional alphabet $\mathfrak{P}$ and restrict ourselves to studying acyclic ProbLog programs in the fragment of ProbLog clauses.

**Convention.** In the following, we restrict the term **ProbLog program** to programs in the fragment of ProbLog clauses that provide full explanation. In fact, we identify such a ProbLog program with a set $\mathbf{P}$ of ProbLog clauses.

Furthermore, we make the assumption that Occam's razor (23) holds, meaning that all programs $\mathbf{P}$ under consideration do not include clauses that are unnecessary for defining the corresponding probability distribution.

**Definition 4.1.1** (Occam's Razor)**.** A ProbLog program $\mathbf{P}$ satisfies **Occam's razor** if every clause $PC \in \mathbf{P}$ has a non-zero probability, i.e. $\pi(PC) \neq 0$, and if the body of every clause $PC \in \mathbf{P}$ has a non-zero probability, i.e. $\pi(\mathrm{body}(PC)) \neq 0$.

**Example 4.1.1.** Consider the following acyclic ProbLog programs in the alphabet $\mathfrak{P} := \{p, q, r, s, t\}$.

$$\begin{array}{lllll}
\mathbf{P}_1 : & 0.5 :: p & 0.5 :: q \leftarrow p & 0.5 :: r \leftarrow \neg p & \\
\mathbf{P}_2 : & 0.5 :: p & 0.5 :: q \leftarrow p & 0.5 :: r \leftarrow \neg p & 0.5 :: s \leftarrow t \\
\mathbf{P}_3 : & 0.5 :: p & 0.5 :: q \leftarrow p & 0.5 :: r \leftarrow \neg p & 0 :: s \\
\mathbf{P}_4 : & 0.5 :: p & 0.5 :: q \leftarrow p & 0.5 :: r \leftarrow \neg p & 0.5 :: s \leftarrow q, r
\end{array}$$

Note that all programs $\mathbf{P}_1 - \mathbf{P}_4$ yield the same distribution. However, program $\mathbf{P}_1$ satisfies Occam's razor, whereas programs $\mathbf{P}_2 - \mathbf{P}_4$ do not.

We aim to specify the families of acyclic ProbLog programs satisfying Occam's razor that yield the same counterfactual probabilities.

**Definition 4.1.2** (Counterfactual Reasoning)**.** A **counterfactual reasoning** $\pi$ on the alphabet $\mathfrak{P}$ consists of a probability $\pi(\phi \mid \mathbf{e}, \mathrm{do}(\mathbf{i})) \in [0, 1]$ for every pair of sets of literals $\mathbf{e}$ and $\mathbf{i}$ and for every formula $\phi$. According to Theorem 3.4.6, every ProbLog program $\mathbf{P}$ providing full explanation gives rise to a counterfactual reasoning $\pi$, which we call the **counterfactual semantics** of $\mathbf{P}$.

**Remark 4.1.1.** We identify a truth value assignment $\mathbf{s}$ on a set of propositions $\mathbf{S} \subseteq \mathfrak{P}$ with the set of literals that it renders true.

Our focus lies on acyclic ProbLog programs that adhere to Occam's razor and exhibit identical counterfactual semantics. Subsequently, we present two examples of such programs.

**Example 4.1.2.** Consider the ProbLog programs $\mathbf{P}_{1/2}$ in the alphabet $\mathfrak{P} := \{a\}$.

$$\begin{array}{lll} \mathbf{P}_1: & 0.5 :: a & 0.6 :: a \\ \mathbf{P}_2: & 0.8 :: a & \end{array}$$

The programs $\mathbf{P}_{1/2}$ are acyclic and satisfy Occam's razor. Since there are only the interventions $\emptyset$, $\{a\}$ and $\{\neg a\}$, we further obtain that the programs $\mathbf{P}_{1/2}$ have the same counterfactual semantics if they yield the same distribution, which is indeed the case.

Example 4.1.2 motivates that we restrict ourselves to ProbLog programs in normal form.

**Definition 4.1.3** (Normal Form)**.** A ProbLog program $\mathbf{P}$ is in **normal form** if any two distinct ProbLog clauses $PC_{1/2} \in \mathbf{P}$ differ either in their heads or bodies, i.e., $\text{head}(PC_1) \neq \text{head}(PC_2)$ or $\text{body}(PC_1) \neq \text{body}(PC_2)$.

**Example 4.1.3.** The program $\mathbf{P}_1$ in Example 4.1.2 is not in normal form, whereas $\mathbf{P}_2$ is.

Fortunately, it is possible to translate every ProbLog program $\mathbf{P}$ into an equivalent ProbLog program in normal form.

**Proposition 4.1.1.** *We can translate every ProbLog program $\boldsymbol{P}$ that is not in normal form to a program $\boldsymbol{P}'$ in normal form with the same counterfactual semantics as $\boldsymbol{P}$.*

*Proof.* Let $\mathbf{P}'$ be the program resulting from $\mathbf{P}$ by replacing all clauses $PC_1, ..., PC_n$ sharing the same head $\text{head}(PC)$ and body $\text{body}(PC)$ with a ProbLog clause

$$\left(1 - \prod_{i=1}^{n}(1 - \pi(PC_i))\right) :: \text{head}(PC) \leftarrow \mathit{body}(PC).$$

According to Theorem 3.4.8, for an intervention $\mathbf{i}$, the counterfactual probabilities $\pi(\phi \mid \mathbf{e}, \text{do}(\mathbf{i}))$ depend only on the probability distribution on the logic programs $\text{LP}(\mathbf{P}_{\mathbf{i}}^{T}) \cup \epsilon$ induced by the choices $\epsilon$. Consequently, $\mathbf{P}$ and $\mathbf{P}'$ yield the same counterfactual semantics. □

Finally, we adapt an example credited to Lifshitz (10, Example 8.3) and derive two additional ProbLog programs that share the same counterfactual semantics.

**Example 4.1.4.** Consider the following acyclic ProbLog programs in normal form, adhering to Occam's razor, within the alphabet $\mathfrak{P} := \{p, q\}$.

$$\begin{array}{llll} \mathbf{P}_1: & 0.5 :: p & 1 :: q \leftarrow p & 1 :: q \leftarrow \neg p \\ \mathbf{P}_2: & 0.5 :: p & 1 :: q & \end{array}$$

Computing all counterfactual probabilities verifies that the programs $\mathbf{P}_{1/2}$ give rise to the same counterfactual semantics.

Because of Example 4.1.4, we further restrict ourselves to acyclic proper ProbLog programs in normal form that satisfy Occam's razor.

**Definition 4.1.4** (Properness)**.** A ProbLog program $\mathbf{P}$ is **proper** if it does not mention trivial probabilities of zero and one, i.e. for every ProbLog clause $PC \in \mathbf{P}$ we find $\pi(PC) \neq 0, 1$.

**Example 4.1.5.** Recall the ProbLog program $\mathbf{P}$ in Example 2.2.48.

$$\begin{array}{ll} 0.5 :: \mathit{cloudy} & \\ 0.6 :: \mathit{rain} \leftarrow \mathit{cloudy} & \\ 0.7 :: \mathit{sprinkler} \leftarrow \neg \mathit{cloudy} & 0.1 :: \mathit{sprinkler} \leftarrow \mathit{cloudy} \\ 0.8 :: \mathit{wet} \leftarrow \mathit{rain} & 0.3 :: \mathit{wet} \leftarrow \mathit{sprinkler} \\ 0.9 :: \mathit{slippery} \leftarrow \mathit{wet} & \end{array}$$

The program $\mathbf{P}$ is proper, whereas the program in Example 4.1.4 is not.

In the upcoming section, we demonstrate that counterfactual reasoning uniquely determines an acyclic proper ProbLog program $\mathbf{P}$ in normal form, satisfying Occam's razor up to syntactic equality.

## 4.2 Knowledge Uniquely Determined by Counterfactual Reasoning

Let us consider a fixed acyclic proper ProbLog program $\mathbf{P}$ in normal form that satisfies Occam's razor. Now imagine that we have forgotten the details of the program $\mathbf{P}$ and only have access to its counterfactual semantics $\pi$. We aim to demonstrate that the counterfactual reasoning $\pi$ alone is sufficient to reconstruct the program $\mathbf{P}$ up to syntactic equality. Our starting point for this endeavor is Lemma 4.2.1, which assists us in identifying the structural information embedded in the counterfactual reasoning $\pi$.

**Lemma 4.2.1** (Important Identities)**.** *Let $\boldsymbol{P}$ be an acyclic proper ProbLog program, and select a proposition $p \in \mathfrak{P}$ together with two supersets*

$$\mathrm{pa}(p) \subseteq \boldsymbol{S}_1, \boldsymbol{S}_2 \subseteq \mathfrak{P} \setminus \{p\}$$

*of the parents* $\mathrm{pa}(p)$ *of $p$ in the dependence graph* $\mathrm{graph}(\boldsymbol{P})$*, not containing $p$ itself. For any two truth value assignments $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ on $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$, respectively, we obtain the following identities.*

$$\pi(p \mid \mathrm{do}(\boldsymbol{s}_1)) = \pi\left(\bigvee_{\substack{PC \in \boldsymbol{P} \\ \mathrm{head}(PC)=p \\ \mathrm{body}(PC)\subseteq \boldsymbol{s}_1}} u(PC)\right) = 1 - \prod_{\substack{PC \in \boldsymbol{P} \\ \mathrm{head}(PC)=p \\ \mathrm{body}(PC)\subseteq \boldsymbol{s}_1}} (1 - \pi(PC)) \tag{4.1}$$

$$\pi(p \mid \boldsymbol{s}_1, p, \mathrm{do}(\boldsymbol{s}_2)) = \pi\left(\bigvee_{\substack{PC \in \boldsymbol{P} \\ \mathrm{head}(PC)=p \\ \mathrm{body}(PC)\subseteq \boldsymbol{s}_2}} u(PC) \;\middle|\; \bigvee_{\substack{PC \in \boldsymbol{P} \\ \mathrm{head}(PC)=p \\ \mathrm{body}(PC)\subseteq \boldsymbol{s}_1}} u(PC)\right) = \tag{4.2}$$

$$= \begin{cases} 1 - \dfrac{1 - \pi(p \mid \mathrm{do}(\boldsymbol{s}_2))}{\pi(p \mid \mathrm{do}(\boldsymbol{s}_1))} \left(1 - \dfrac{(1 - \pi(p \mid \mathrm{do}(\boldsymbol{s}_1)))}{\prod\limits_{\substack{PC \in \boldsymbol{P} \\ \mathrm{head}(PC)=p \\ \mathrm{body}(PC)\subseteq \boldsymbol{s}_1 \cap \boldsymbol{s}_2}} (1 - \pi\,(PC))}\right) & \text{if } \pi(p, \boldsymbol{s}_1) \neq 0 \\ 0 & \text{else} \end{cases} \tag{4.3}$$

*Proof.* To prove Equation (4.1) we observe

$$\pi(p \mid \mathrm{do}(\mathbf{s}_1)) \overset{\text{choice of } \mathbf{S}_1 \text{ and } \mathbf{S}_2}{=} \pi\left(\bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC)=p \\ \mathrm{body}(PC)\subseteq \mathbf{s}_1}} u(PC)\right) = 1 - \prod_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC)=p \\ \mathrm{body}(PC)\subseteq \mathbf{s}_1}} (1 - \pi(PC)).$$

Further, we turn to the proof of Equation (4.2) and (4.3). If $\pi(p, \mathbf{s}_1) = 0$, Remark 3.3.1 yields $\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) = 0$. Otherwise, we use Theorem 3.4.8 and obtain

$$\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) \overset{\text{choice of } \mathbf{S}_{1/2}}{=} \pi\left( \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = p \\ \mathrm{body}(PC) \subseteq \mathbf{s}_2}} u(PC) \;\middle|\; p \wedge \mathbf{s}_1 \right).$$

For a choice $\epsilon$, the logic program $\mathrm{LP}(\mathbf{P}) \cup \epsilon$ has the supported model $\omega = \{p\} \cup \mathbf{s}_1$ if and only if

$$\epsilon \models \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = p \\ \mathrm{body}(PC) \subseteq \mathbf{s}_1}} u(PC) \wedge \bigwedge_{q \in \mathfrak{P} \cap \mathbf{s}_1} \left( \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = q \\ \mathrm{body}(PC) \subseteq \mathbf{s}_1}} u(PC) \right) \wedge \bigwedge_{\substack{PC \in \mathbf{P} \\ q \in \mathfrak{P} \setminus (\mathbf{s}_1 \cup \{p\}) \\ \mathrm{head}(PC) = q \\ \mathrm{body}(PC) \subseteq \mathbf{s}_1}} \neg u(PC). \tag{4.4}$$

Since $\mathbf{P}$ is acyclic, we obtain from the independence of the error terms that

$$\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) = \pi\left( \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = p \\ \mathrm{body}(PC) \subseteq \mathbf{s}_2}} u(PC) \;\middle|\; \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = p \\ \mathrm{body}(PC) \subseteq \mathbf{s}_1}} u(PC) \right) =$$

$$= 1 - \pi\left( \bigwedge_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = p \\ \mathrm{body}(PC) \subseteq \mathbf{s}_2}} \neg u(PC) \;\middle|\; \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = p \\ \mathrm{body}(PC) \subseteq \mathbf{s}_1}} u(PC) \right) =$$

$$= 1 - \frac{\pi\left( \bigvee_{\substack{PC_1 \in \mathbf{P} \\ \mathrm{head}(PC_1) = p \\ \mathrm{body}(PC_1) \subseteq \mathbf{s}_1}} u(PC_1) \wedge \left( \bigwedge_{\substack{PC_2 \in \mathbf{P} \\ \mathrm{head}(PC_2) = p \\ \mathrm{body}(PC_2) \subseteq \mathbf{s}_2}} \neg u(PC_2) \right) \right)}{\pi\left( \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = p \\ \mathrm{body}(PC) \subseteq \mathbf{s}_1}} u(PC) \right)} \overset{\substack{\text{independence of} \\ \text{error terms \& (4.1)}}}{=}$$

$$= 1 - \frac{1 - \pi(p \mid \mathrm{do}(\mathbf{s}_2))}{\pi(p \mid \mathrm{do}(\mathbf{s}_1))} \cdot \pi\left( \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = p \\ \mathrm{body}(PC) \subseteq \mathbf{s}_1 \\ \mathrm{body}(PC) \not\subseteq \mathbf{s}_2}} u(PC) \right).$$

Again, invoking the independence of the error terms, we deduce that

$$\pi\left(\bigvee_{\substack{PC\in\mathbf{P}\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{s}_1\\ \mathrm{body}(PC)\not\subseteq\mathbf{s}_2}} u(PC)\right) = 1 - \frac{\pi\left(\bigwedge_{\substack{PC\in\mathbf{P}\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{s}_1}} \neg u(PC)\right)}{\pi\left(\bigwedge_{\substack{PC\in\mathbf{P}\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{s}_1\cap\mathbf{s}_2}} \neg u(PC)\right)} = 1 - \frac{1-\pi(p\mid \mathrm{do}(\mathbf{s}_1))}{\prod_{\substack{PC\in\mathbf{P}\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{s}_1\cap\mathbf{s}_2}} (1-\pi(PC))}.$$

□

The next proposition explicates the structural information about **P** uncovered by Lemma 4.2.1.

**Proposition 4.2.2** (Structural Information in Counterfactual Semantics)**.** *Recall the context of Lemma 4.2.1.*

*i) We find* $\pi(p \mid \boldsymbol{s}_1, p, \mathrm{do}(\boldsymbol{s}_2)) = 1$ *if and only if* $\pi(\boldsymbol{s}_1, p) > 0$ *and for every clause* $PC \in \boldsymbol{P}$ *with* $\mathrm{head}(PC) = p$ *and* $\mathrm{body}(PC) \subseteq \boldsymbol{s}_1$*, we obtain* $\mathrm{body}(PC) \subseteq \boldsymbol{s}_2$*.*

*ii) We find* $\pi(p \mid \boldsymbol{s}_1, p, \mathrm{do}(\boldsymbol{s}_2)) > \pi(p \mid \mathrm{do}(\boldsymbol{s}_2))$ *if and only if* $\pi(p, \boldsymbol{s}_1) > 0$ *and there exists a clause* $PC \in \boldsymbol{P}$ *with* $\mathrm{body}(PC) \subseteq \boldsymbol{s}_1 \cap \boldsymbol{s}_2$*.*

*Proof.* We begin by proving i). Assume that we find $\pi(\mathbf{s}_1, p) > 0$ and $\mathrm{body}(PC) \subseteq \mathbf{s}_2$ for every clause $PC \in \mathbf{P}$ with $\mathrm{head}(PC) = p$ and $\mathrm{body}(PC) \subseteq \mathbf{s}_1$. In this case, $\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) = 1$ follows from Equations (4.1) and (4.3) of Lemma 4.2.1. Further, assume $\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) = 1$. In particular, we find $\pi(p, \mathbf{s}_1) > 0$, and by properness, we have $\pi(p \mid \mathrm{do}(\mathbf{s}_2)) < 1$. Hence, Equations (4.1) and (4.3) yield

$$\prod_{\substack{PC\in\mathbf{P}\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{s}_1}} (1-\pi(PC)) = 1 - \pi(p \mid \mathrm{do}(\mathbf{s}_1)) = \prod_{\substack{PC\in\mathbf{P}\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{s}_1\cap\mathbf{s}_2}} (1-\pi\,(PC)).$$

Finally, from properness we deduce that $\mathrm{body}(PC) \subseteq \mathbf{s}_2$ for every ProbLog clause $PC \in \mathbf{P}$ with $\mathrm{head}(PC) = p$ and $\mathrm{body}(PC) \subseteq \mathbf{s}_1$.

Next, we prove ii). Assume $1 \geq \pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) > \pi(p \mid \mathrm{do}(\mathbf{s}_2)) \geq 0$. Again, we find $\pi(p, \mathbf{s}_1) > 0$ and $\pi(p \mid \mathrm{do}(\mathbf{s}_2)) < 1$. Hence, Equations (4.1) and (4.3) ensure that

$$\prod_{\substack{PC\in\mathbf{P}\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{s}_1\cap\mathbf{s}_2}} (1-\pi(PC)) < 1 \tag{4.5}$$

and we obtain a clause $PC \in \mathbf{P}$ such that $\mathrm{body}(PC) \subseteq \mathbf{s}_1 \cap \mathbf{s}_2$. Finally, assume $\pi(p, \mathbf{s}_1) > 0$ and $\mathrm{body}(PC) \subseteq \mathbf{s}_1 \cap \mathbf{s}_2$ for a clause $PC \in \mathbf{P}$ with $\mathrm{head}(PC) = p$. In this case, we find that Inequality (4.5) and Equation (4.3) yield $\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) > \pi(p \mid \mathrm{do}(\mathbf{s}_2))$. □

Furthermore, we demonstrate the necessity of the acyclicity assumption in Lemma 4.2.1 and Proposition 4.2.2 i).

**Example 4.2.1.** Consider the following proper ProbLog program that satisfies Occam's razor.

$$\mathbf{P:} \qquad 0.5 :: a \leftarrow b \qquad 0.5 :: b \leftarrow a \qquad 0.5 :: a.$$

In this case, we find $\pi(a \mid a, b, \mathrm{do}(\neg b)) = 1$ even though $PC := (0.5 :: a \leftarrow b) \in \mathbf{P}$ yields a clause with $\mathrm{body}(PC) = \{b\}$, contradicting statement i) of Proposition 4.2.2.

Let us now begin with reconstructing the dependence graph graph($\mathbf{P}$) from the counterfactual semantics $\pi$ of $\mathbf{P}$. For a more readable presentation, we introduce the following notions.

**Definition 4.2.1** (Situations and Frames)**.** A **situation** for a proposition $p \in \mathfrak{P}$ is a truth value assignment $\mathbf{s}$ to the remaining propositions in $\mathfrak{P} \setminus \{p\}$. For two situations $\mathbf{s}_{1/2}$ we call the number $\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) \in [0,1]$ the **change of situations** from $\mathbf{s}_1$ to $\mathbf{s}_2$. Furthermore, the **common support** of $\mathbf{s}_1$ and $\mathbf{s}_2$ is defined by $\mathrm{supp}(\mathbf{s}_1, \mathbf{s}_2) := \{p \in \mathfrak{P} : p^{\mathbf{s}_1} = p^{\mathbf{s}_2}\}$ .

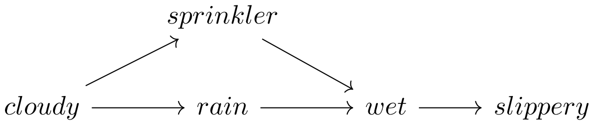
Given the dependence graph graph($\mathbf{P}$) of $\mathbf{P}$ we call a truth value assignment $\mathbf{f}$ to the parents pa($p$) of a proposition $p \in \mathfrak{P}$ a **frame** for $p$. Finally, for two frames $\mathbf{f}_{1/2}$ we call the number $\pi(p \mid \mathbf{f}_1, p, \mathrm{do}(\mathbf{f}_2)) \in [0,1]$ the **reframing** from $\mathbf{f}_1$ to $\mathbf{f}_2$.

We compute the dependence graph using the following procedure:

**Procedure 4.2.1** (Reconstructing the Dependence Graph)**.** Let $p \in \mathfrak{P}$ be a proposition. If we find $\pi(p, \mathbf{s}) = 0$ for all situations $\mathbf{s}$, i.e. $\pi(p) = 0$, we set $\mathrm{pa}_G(p) = \emptyset$. Otherwise, we compute the smallest set $\mathrm{pa}_G(p) \subseteq \mathfrak{P} \setminus \{p\}$ such that for every two situations $\mathbf{s}_{1/2}$ with $\pi(p, \mathbf{s}_1) > 0$ we find $\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) = 1$ and $\pi(p \mid \mathbf{s}_2, p, \mathrm{do}(\mathbf{s}_1)) = 1$ whenever $\mathrm{pa}_G(p) \subseteq \mathrm{supp}(\mathbf{s}_1, \mathbf{s}_2)$. Finally, we construct the graph $G$ by drawing an arrow from every node in $\mathrm{pa}_G(p)$ to the node $p$.

**Example 4.2.2.** Recall the program $\mathbf{P}$ with the dependence graph graph($\mathbf{P}$) in Example 2.2.48.

$$\begin{array}{ll}
0.5 :: \mathit{cloudy} & \\
0.6 :: \mathit{rain} \leftarrow \mathit{cloudy} & \\
0.7 :: \mathit{sprinkler} \leftarrow \neg \mathit{cloudy} & 0.1 :: \mathit{sprinkler} \leftarrow \mathit{cloudy} \\
0.8 :: \mathit{wet} \leftarrow \mathit{rain} & 0.3 :: \mathit{wet} \leftarrow \mathit{sprinkler} \\
0.9 :: \mathit{slippery} \leftarrow \mathit{wet} &
\end{array}$$



Assume we aim to find the parents $\mathrm{pa}_{\mathrm{graph}(\mathbf{P})}(\mathit{slippery}) = \{\mathit{wet}\}$ of *slippery* in the dependence graph graph($\mathbf{P}$) of the program $\mathbf{P}$. Initially, we discover that *slippery* is only true when it is wet, which occurs if it rains or if the sprinkler is on. Consequently, we consider the following situations.

$$\begin{array}{ll}
\mathbf{s}_1 := \{\mathit{cloudy}, \mathit{rain}, \mathit{sprinkler}, \mathit{wet}\} & \mathbf{s}_2 := \{\neg\mathit{cloudy}, \mathit{rain}, \mathit{sprinkler}, \mathit{wet}\} \\
\mathbf{s}_3 := \{\mathit{cloudy}, \neg\mathit{rain}, \mathit{sprinkler}, \mathit{wet}\} & \mathbf{s}_4 := \{\mathit{cloudy}, \mathit{rain}, \neg\mathit{sprinkler}, \mathit{wet}\} \\
\mathbf{s}_5 := \{\neg\mathit{cloudy}, \neg\mathit{rain}, \mathit{sprinkler}, \mathit{wet}\} & \mathbf{s}_6 := \{\neg\mathit{cloudy}, \mathit{rain}, \neg\mathit{sprinkler}, \mathit{wet}\}
\end{array}$$

Upon examination, we note that $\pi(\mathit{slippery} \mid \mathbf{s}_i, \mathit{slippery}, \mathrm{do}(\mathbf{s}_i)) = 1$ for all $1 \leq i \leq 6$, as all relevant error terms were already observed to be true. Consequently, we investigate the 30 changes of situations $\pi(\mathit{slippery} \mid \mathbf{s}_i, \mathit{slippery}, \mathrm{do}(\mathbf{s}_j))$ for $1 \leq i, j \leq 6$, $i \neq j$. The WhatIf solver (34) then confirms that $\pi(\mathit{slippery} \mid \mathbf{s}_i, \mathit{slippery}, \mathrm{do}(\mathbf{s}_j)) = 1$ for every change of situation under consideration. We observe that all the situations $\mathbf{s}_1$-$\mathbf{s}_6$ only coincide in *wet*, leading to

$$\mathrm{pa}_G(\mathit{slippery}) = \{\mathit{wet}\} = \mathrm{pa}_{\mathrm{graph}(\mathbf{P})}(\mathit{slippery})$$

as the set of parents for *slippery* in the graph $G$ from Procedure 4.2.1.

Finally, we prove that Procedure 4.2.1 actually computes the dependence graph graph($\mathbf{P}$) of $\mathbf{P}$.

**Lemma 4.2.3.** *Let $\boldsymbol{P}$ be an acyclic proper ProbLog program that satisfies Occam's razor. In this case, Procedure 4.2.1 indeed computes the dependence graph $G = \mathrm{graph}(\boldsymbol{P})$ of $\boldsymbol{P}$.*

*Proof.* Let $p \in \mathfrak{P}$ be a proposition and denote by $\mathrm{pa}_G(p)$ the parents of $p$ in $G$ and by $\mathrm{pa}_{\mathrm{graph}(\mathbf{P})}(p)$ the parents of $p$ in the dependence graph $\mathrm{graph}(\mathbf{P})$. We show $\mathrm{pa}_G(p) = \mathrm{pa}_{\mathrm{graph}(\mathbf{P})}(p)$. If $\pi(p) = 0$, we find that $\pi(p, \mathbf{s}) = 0$ for all situations $\mathbf{s}$. Hence, by Occam's razor, there is no clause $PC \in \mathbf{P}$ with $\mathrm{head}(PC) = p$, and we obtain $\mathrm{pa}_G(p) = \emptyset = \mathrm{pa}_{\mathrm{graph}(\mathbf{P})}(p)$.

Further, assume $\pi(p) > 0$. If two situations $\mathbf{s}_{1/2}$ with $\pi(p, \mathbf{s}_1) > 0$ coincide on the parents $\mathrm{pa}_{\mathrm{graph}(\mathbf{P})}(p)$ of $p$ in the dependence graph, by Definition 2.2.33, we have $\mathrm{body}(PC) \subseteq \mathbf{s}_1$ if and only if $\mathrm{body}(PC) \subseteq \mathbf{s}_2$ for every clause $PC \in \mathbf{P}$ with $\mathrm{head}(PC) = p$. Hence, Proposition 4.2.2 yields $\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) = 1$ as well as $\pi(p \mid \mathbf{s}_2, p, \mathrm{do}(\mathbf{s}_1)) = 1$, and we obtain $\mathrm{pa}_G(p) \subseteq \mathrm{pa}_{\mathrm{graph}(\mathbf{P})}(p)$.

Next, we need to show that $\mathrm{pa}_G(p) \supseteq \mathrm{pa}_{\mathrm{graph}(\mathbf{P})}(p)$. Let $b \in \mathrm{pa}_{\mathrm{graph}(\mathbf{P})}(p)$. By Definition 2.2.33, there exists a clause $PC \in \mathbf{P}$ with $\{b, \neg b\} \cap \mathrm{body}(PC) \neq \emptyset$. According to Occam's razor, $\mathrm{body}(PC)$ extends to a situation $\mathbf{s}_1$ with $\pi(p, \mathbf{s}_1) \neq 0$. Set $\mathbf{s}_2 := (\mathbf{s}_1 \setminus \{b, \neg b\}) \cup (\{b, \neg b\} \setminus \mathbf{s}_1)$, i.e., the situations $\mathbf{s}_{1/2}$ only differ in the truth value for the proposition $b \in \mathfrak{P}$. Finally, we apply Proposition 4.2.2 to obtain $\pi(p \mid \mathbf{s}_1, p, \mathrm{do}(\mathbf{s}_2)) < 1$. Hence, from the construction of $\mathbf{s}_2$, we may conclude that $b \in \mathrm{pa}_G(p)$ as desired. □

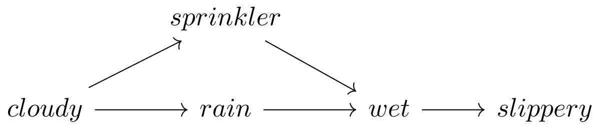
With the dependence graph available, our next goal is reconstructing the clauses in the program $\mathbf{P}$. Let us consider a proposition $p \in \mathfrak{P}$ to compute the clauses defining $p$. We begin with the following notion.

**Definition 4.2.2** (Clause Search Graph)**.** The **clause search graph** $\mathrm{Search}(p)$ of $p$ is an undirected graph on the frames $\mathbf{f}$ of $p$ with $\pi(p, \mathbf{f}) > 0$. It is given by drawing an edge $\mathbf{f}_1 - \mathbf{f}_2$ if and only if $\pi(p \mid \mathbf{f}_1, p, \mathrm{do}(\mathbf{f}_2)) > \pi(p \mid \mathrm{do}(\mathbf{f}_2))$. Finally, we equip each edge $\mathbf{f}_1 - \mathbf{f}_2$ with the **label** $\mathbf{f}_1 \cap \mathbf{f}_2$.

**Remark 4.2.1.** Since $\mathbf{P}$ satisfies Occam's razor, we do not find any clause $PC \in \mathbf{P}$ such that $\mathrm{body}(PC) \subseteq \mathbf{f}$ for a frame $\mathbf{f}$ with $\pi(p, \mathbf{f}) = 0$.

**Example 4.2.3.** Recall the program $\mathbf{P}$ with the dependence graph $\mathrm{graph}(\mathbf{P})$ in Example 2.2.48.

$$\begin{array}{ll}
0.5 :: \mathit{cloudy} & \\
0.6 :: \mathit{rain} \leftarrow \mathit{cloudy} & \\
0.7 :: \mathit{sprinkler} \leftarrow \neg \mathit{cloudy} & 0.1 :: \mathit{sprinkler} \leftarrow \mathit{cloudy} \\
0.8 :: \mathit{wet} \leftarrow \mathit{rain} & 0.3 :: \mathit{wet} \leftarrow \mathit{sprinkler} \\
0.9 :: \mathit{slippery} \leftarrow \mathit{wet} &
\end{array}$$



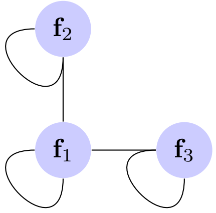
Assume we already know the dependence graph $\mathrm{graph}(\mathbf{P})$ of the program $\mathbf{P}$ and want to recover the clauses defining *wet*. Note that there are three frames $\mathbf{f}$ with $\pi(\mathit{wet}, \mathbf{f}) > 0$:

$$\mathbf{f}_1 := \{\mathit{rain}, \mathit{sprinkler}\} \qquad \mathbf{f}_2 := \{\neg \mathit{rain}, \mathit{sprinkler}\} \qquad \mathbf{f}_3 := \{\mathit{rain}, \neg \mathit{sprinkler}\}$$

The WhatIf solver (34) yields $\pi(\mathit{wet} \mid \mathbf{f}_i, \mathit{wet}\, \mathrm{do}(\mathbf{f}_i)) = 1 > \pi(\mathit{wet} \mid \mathrm{do}(\mathbf{f}_i))$ for $1 \leq i \leq 3$, and we obtain

$$\begin{aligned}
\pi(\mathit{wet} \mid \mathbf{f}_1, \mathit{wet}, \mathrm{do}(\mathbf{f}_2)) &= 0.3488... > 0.3 = \pi(\mathit{wet} \mid \mathbf{f}_2) \\
\pi(\mathit{wet} \mid \mathbf{f}_1, \mathit{wet}, \mathrm{do}(\mathbf{f}_3)) &= 0.9302... > 0.8 = \pi(\mathit{wet} \mid \mathbf{f}_3) \\
\pi(\mathit{wet} \mid \mathbf{f}_2, \mathit{wet}, \mathrm{do}(\mathbf{f}_3)) &= 0.8 = \pi(\mathit{wet} \mid \mathbf{f}_3).
\end{aligned}$$

This yields the following clause search graph $\mathrm{Search}(\mathit{wet})$.



Now that we have the clause search graph, let us uncover the first clause defining $p$, which we denote as $PC_{0,p}$.

**Lemma 4.2.4** (Finding a Clause)**.** *Let $\boldsymbol{P}$ be an acyclic proper ProbLog program in normal form. Let $\mathrm{body}_{0,p} = \boldsymbol{f}_1 \cap \boldsymbol{f}_2$ be a minimal label of an edge $\boldsymbol{f}_1 - \boldsymbol{f}_2$ in the clause search graph $\mathrm{Search}(p)$ of a proposition $p \in \mathfrak{P}$, that is, there exists no edge $\boldsymbol{f}_1' - \boldsymbol{f}_2'$ with label $\boldsymbol{f}_1' \cap \boldsymbol{f}_2' \subsetneq \mathrm{body}_{0,p}$. In this case, we find a unique clause $PC_{0,p} \in \boldsymbol{P}$ with $\mathrm{head}(PC_{0,p}) = p$, with $\mathrm{body}(PC_{0,p}) = \mathrm{body}_{0,p}$ and with probability*

$$\pi(PC_{0,p}) = 1 - \frac{(1 - \pi(p \mid \mathrm{do}(\boldsymbol{f}_1))) \cdot (1 - \pi(p \mid \mathrm{do}(\boldsymbol{f}_2)))}{1 - \pi(p \mid \mathrm{do}(\boldsymbol{f}_2)) - \pi(p \mid \mathrm{do}(\boldsymbol{f}_1)) \cdot (1 - \pi(p \mid \boldsymbol{f}_1, p, \mathrm{do}(\boldsymbol{f}_2)))}. \tag{4.6}$$

*Proof.* According to Proposition 4.2.2 and Definition 4.2.2, we find a clause $PC_{0,p} \in \mathbf{P}$ such that

$$\mathrm{body}(PC_{0,p}) \subseteq \mathrm{body}_{0,p}\,.$$

If we assume that $\mathrm{body}(PC_{0,p}) \neq \mathrm{body}_{0,p}$, by minimality of $\mathrm{body}_{0,p}$, this clause would induce an edge not present in the clause search graph $\mathrm{Search}(p)$. Since $\mathbf{P}$ is assumed to be in normal form, we further obtain that $PC_{0,p} \in \mathbf{P}$ is the unique clause with $\mathrm{body}(PC_{0,p}) = \mathrm{body}_{0,p}$ and with $\mathrm{head}(PC_{0,p}) = p$. Finally, consider Equation (4.3) of Lemma 4.2.1. This allows us to see that

$$\pi(p \mid \mathbf{f}_1, p, \mathrm{do}(\mathbf{f}_2)) = 1 - \frac{1 - \pi(p \mid \mathrm{do}(\mathbf{f}_2))}{\pi(p \mid \mathrm{do}(\mathbf{f}_1))} \left(1 - \frac{1 - \pi(p \mid \mathrm{do}(\mathbf{f}_1))}{1 - \pi(PC_{0,p})}\right).$$

From this, solving for $\pi(PC_{0,p})$ yields the desired result. □

**Example 4.2.4.** In Example 4.2.3, we find that $\mathbf{f}_1 - \mathbf{f}_2$ yields an edge of the clause search graph $\mathrm{Search}(wet)$ with minimal label $\mathrm{body}_{0,wet} := \mathbf{f}_1 \cap \mathbf{f}_2 = \{sprinkler\}$. Further, Equation (4.6) and a calculation with the WhatIf solver (34) yield that

$$\pi(PC_{0,wet}) := 1 - \frac{0.14 \cdot 0.7}{1 - 0.3 - 0.86 \cdot (1 - 0.3488...)} = 0.3.$$

Overall, we found the clause $PC_{0,wet} := (0.3 :: wet \leftarrow sprinkler)$ that indeed appears in the program $\mathbf{P}$ of Example 4.2.3. .

To proceed, we observe that the whole counterfactual reasoning $\pi$ is not necessary to reconstruct the clause $PC_{0,p} \in \mathbf{P}$. In fact, we only require the data provided by a counterfactual backbone.

**Definition 4.2.3** (Counterfactual Backbone)**.** A **counterfactual backbone** (of a ProbLog program $\mathbf{P}$) comprises the probabilities $\pi(p \mid \mathrm{do}(\mathbf{f}_{1/2}))$ and the reframings $\pi(p \mid \mathbf{f}_1, p, \mathrm{do}(\mathbf{f}_2))$ for all frames $\mathbf{f}_{1/2}$ of a proposition $p \in \mathfrak{P}$.

**Example 4.2.5.** The counterfactual backbone of the program $\mathbf{P}$ in Example 4.2.3 has the following form.

$$\begin{aligned}
&\pi(\mathit{cloudy}) = 0.5,\ \pi(\mathit{sprinkler} \mid \mathrm{do}(\mathit{cloudy})) = 0.1,\ \pi(\mathit{sprinkler} \mid \mathrm{do}(\neg \mathit{cloudy})) = 0.7,\\
&\pi(\mathit{rain} \mid \mathrm{do}(\mathit{cloudy})) = 0.6,\ \pi(\mathit{rain} \mid \mathrm{do}(\neg \mathit{cloudy})) = 0,\\
&\pi(\mathit{wet} \mid \mathrm{do}(\mathit{rain}, \mathit{sprinkler})) = 0.86,\dots,\ \pi(\mathit{wet} \mid \mathrm{do}(\neg \mathit{rain}, \neg \mathit{sprinkler})) = 0,\\
&\pi(\mathit{slippery} \mid \mathrm{do}(\mathit{wet})) = 0.9,\ \pi(\mathit{slippery} \mid \mathrm{do}(\neg \mathit{wet})) = 0,\\
&\pi(\mathit{sprinkler} \mid \mathit{cloudy}, \mathit{sprinkler}, \mathrm{do}(\neg \mathit{cloudy})) = 0.7,\ \pi(\mathit{rain} \mid \mathit{cloudy}, \mathit{rain}, \mathrm{do}(\neg \mathit{cloudy})) = 0,\\
&\pi(\mathit{sprinkler} \mid \neg \mathit{cloudy}, \mathit{sprinkler}, \mathrm{do}(\mathit{cloudy})) = 0.1,\ \pi(\mathit{rain} \mid \neg \mathit{cloudy}, \mathit{rain}, \mathrm{do}(\mathit{cloudy})) = 0,\\
&\pi(\mathit{wet} \mid \mathit{sprinkler}, \mathit{rain}, \mathit{wet}, \mathrm{do}(\neg \mathit{sprinkler}, \neg \mathit{rain})) = 0,\\
&\pi(\mathit{wet} \mid \mathit{sprinkler}, \mathit{rain}, \mathit{wet}, \mathrm{do}(\mathit{sprinkler}, \neg \mathit{rain})) = 0.348\dots,\ \dots,\\
&\pi(\mathit{wet} \mid \neg \mathit{sprinkler}, \mathit{rain}, \mathit{wet}, \mathrm{do}(\mathit{sprinkler}, \mathit{rain})) = 1,\\
&\pi(\mathit{splippery} \mid \mathit{wet}, \mathit{splippery}, \mathrm{do}(\neg \mathit{wet})) = 0,\ \pi(\mathit{splippery} \mid \neg \mathit{wet}, \mathit{splippery}, \mathrm{do}(\mathit{wet})) = 0
\end{aligned}$$

Further, we need the following result to compute the counterfactual backbone of the program $\mathbf{P}_{1,p} := \mathbf{P} \setminus \{PC_{0,p}\}$.

**Lemma 4.2.5** (Modularity of Counterfactual Backbones)**.** *Let $\boldsymbol{P}$ be an acyclic proper ProbLog program, and consider a clause $PC_0 \in \boldsymbol{P}$ with $\mathrm{head}(PC_0) := p \in \mathfrak{P}$. Additionally, fix frames $\boldsymbol{f}_{1/2}$ for $p$ such that $\pi(p, \boldsymbol{f}_{1/2}) > 0$. We denote the counterfactual backbone of the program $\boldsymbol{P}_0 := \boldsymbol{P} \setminus \{PC_0\}$ by $\pi_0$. If $\mathrm{body}(PC_0) \not\subseteq \boldsymbol{f}_i$, we obtain*

$$\pi_0(p \mid \mathrm{do}(\boldsymbol{f}_i)) = \pi(p \mid \mathrm{do}(\boldsymbol{f}_i)). \tag{4.7}$$

*Otherwise, we obtain*

$$\pi_0(p \mid \mathrm{do}(\boldsymbol{f}_i)) = \frac{\pi(p \mid \mathrm{do}(\boldsymbol{f}_i)) - \pi(PC_0)}{1 - \pi(PC_0)}. \tag{4.8}$$

*If $\mathrm{body}(PC_0) \not\subseteq \boldsymbol{f}_1$ and $\mathrm{body}(PC_0) \not\subseteq \boldsymbol{f}_2$, we obtain*

$$\pi_0(p \mid \boldsymbol{f}_1, p, \mathrm{do}(\boldsymbol{f}_2)) = \pi(p \mid \boldsymbol{f}_1, p, \mathrm{do}(\boldsymbol{f}_2)). \tag{4.9}$$

*If $\mathrm{body}(PC_0) \subseteq \boldsymbol{f}_1 \cap \boldsymbol{f}_2$, we obtain*

$$\pi_0(p \mid \boldsymbol{f}_1, p, \mathrm{do}(\boldsymbol{f}_2)) = \pi(p \mid \boldsymbol{f}_1, p, \mathrm{do}(\boldsymbol{f}_2)) - \frac{1 - \pi(p \mid \boldsymbol{f}_1, p, \mathrm{do}(\boldsymbol{f}_2))}{\pi(p \mid \mathrm{do}(\boldsymbol{f}_1)) - \pi(PC_0)} \cdot \pi(PC_0). \tag{4.10}$$

*Further, assume that $\mathrm{body}(PC_0) \subseteq \boldsymbol{f}_2$ while $\mathrm{body}(PC_0) \not\subseteq \boldsymbol{f}_1$. We obtain*

$$\pi_0(p \mid \boldsymbol{f}_1, p, \mathrm{do}(\boldsymbol{f}_2)) = \frac{\pi(p \mid \boldsymbol{f}_1, p, \mathrm{do}(\boldsymbol{f}_2)) - \pi(PC_0)}{1 - \pi(PC_0)}. \tag{4.11}$$

*Finally, assume that $\mathrm{body}(PC_0) \not\subseteq \boldsymbol{f}_2$ while $\mathrm{body}(PC_0) \subseteq \boldsymbol{f}_1$. We obtain*

$$\pi_0(p \mid \boldsymbol{f}_1, p, \mathrm{do}(\boldsymbol{f}_2)) = \frac{\pi(p \mid \boldsymbol{f}_2, p, \mathrm{do}(\boldsymbol{f}_1)) - \pi(PC_0)}{\pi(p \mid \mathrm{do}(\boldsymbol{f}_1)) - \pi(PC_0)} \cdot \pi(p \mid \mathrm{do}(\boldsymbol{f}_2)). \tag{4.12}$$

*Proof.* Equations (4.7) and (4.9) hold trivially, since $PC_0$ is not applicable in either $\mathbf{f}_1$ or $\mathbf{f}_2$. For Equation (4.8) we calculate

$$\pi(p \mid \mathrm{do}(\mathbf{f}_i)) = \pi\left(\bigvee_{\substack{PC\in\mathbf{P}\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_i}} u(PC) \vee u(PC_0)\right) =$$

$$= \pi\left(\bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_i}} u(PC)\right) + \pi(PC_0) - \pi(PC_0)\pi\left(\bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_i}} u(PC)\right) \overset{(4.1)}{=}$$

$$= \pi_0\left(p \mid \mathrm{do}(\mathbf{f}_i)\right) + \pi(PC_0) - \pi(PC_0)\pi_0\left(p \mid \mathrm{do}(\mathbf{f}_i)\right)$$

and solve for $\pi_0\left(p \mid \mathrm{do}(\mathbf{f}_i)\right)$. For Equation (4.10) we calculate

$$\pi(p \mid \mathbf{f}_1, p, \mathrm{do}(\mathbf{f}_2)) \overset{(4.2)}{=}$$

$$= \pi\left(\bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_2}} u(PC) \vee u(PC_0) \;\middle|\; \bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_1}} u(PC) \vee u(PC_0)\right) \overset{\substack{\text{Def. of cond.}\\ \text{prob.}}}{=}$$

$$= \frac{\pi\left(u(PC_0) \vee \left(\bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_2}} u(PC) \wedge \bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_1}} u(PC)\right)\right)}{\pi\left(\bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_1}} u(PC) \vee u(PC_0)\right)} = t_1 \times t_2$$

where we find

$$t_1 := \frac{\pi\left(u(PC_0) \vee \left(\bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_2}} u(PC) \wedge \bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_1}} u(PC)\right)\right)}{\pi_0(p \mid \mathrm{do}(\mathbf{f}_1))}$$

$$t_2 := \frac{\pi_0(p \mid \mathrm{do}(\mathbf{f}_1))}{\pi\left(\bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_1}} u(PC) \vee u(PC_0)\right)}.$$

Expanding the outer $\vee$ and repeatedly applying Equation (4.1) and (4.3) yields

$$t_1 = \frac{\pi(PC_0)}{\pi_0\left(p \mid \mathrm{do}(\mathbf{f}_1)\right)} + \pi_0\left(p \mid \mathbf{f}_1, p, \mathrm{do}(\mathbf{f}_2)\right)\left(1 - \pi(PC_0)\right)$$
$$t_2 = \frac{\pi_0\left(p \mid \mathrm{do}(\mathbf{f}_1)\right)}{\pi_0\left(p \mid \mathrm{do}(\mathbf{f}_1)\right)\left(1 - \pi(PC_0)\right) + \pi(PC_0)}.$$

Now, solving for $\pi_0\left(p \mid \mathbf{f}_1, p, \mathrm{do}(\mathbf{f}_2)\right)$ and Equation (4.1) yield Equation (4.10).

For Equation (4.11) we expand the $\vee$ before $u(PC_0)$ in

$$\pi(p \mid \mathbf{f}_1, p, \mathrm{do}(\mathbf{f}_2)) = \pi\left(\bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_2}} u(PC) \vee u(PC_0) \;\middle|\; \bigvee_{\substack{PC\in\mathbf{P}_0\\ \mathrm{head}(PC)=p\\ \mathrm{body}(PC)\subseteq\mathbf{f}_1}} u(PC)\right)$$

and repeatedly apply Equation (4.1) and (4.3). Finally, Equation (4.12) follows from Equation (4.11) with the identity

$$\pi_0(p \mid \mathbf{f}_1, p, \mathrm{do}(\mathbf{f}_2)) \overset{(4.1)\&(4.2)}{=} \frac{\pi_0(p \mid \mathbf{f}_2, p, \mathrm{do}(\mathbf{f}_1))\pi_0(p \mid \mathrm{do}(\mathbf{f}_2))}{\pi_0(p \mid \mathrm{do}(\mathbf{f}_1))}.$$

□

With Lemma 4.2.5 at hand, we now recover the whole program $\mathbf{P}$ with the following procedure.

**Procedure 4.2.2.** We start with a counterfactual reasoning or a counterfactual backbone $\pi$, which is known to stem from the counterfactual semantics of a hidden acyclic proper ProbLog program $\mathbf{P}$ in normal form that satisfies Occam's razor. In addition, we assume that we are given the dependence graph graph($\mathbf{P}$) of $\mathbf{P}$. We compute the ProbLog program $\tilde{\mathbf{P}}$ according to the following algorithm.

```
$\tilde{\mathbf{P}} := \emptyset$
**for** $p \in \mathfrak{P}$ **do**
    $\tilde{\mathbf{P}}_p := \emptyset$; $i := 0$; $\pi_{0,p} := \pi$;
    **while** $\pi_{i,p}(p| \mathrm{do}(\mathbf{f})) \neq 0$ for a frame $\mathbf{f}$ **do**
        Compute the clause search graph $\mathrm{Search}_i(p)$ according to the counterfactual backbone $\pi_{i,p}$
        Choose edge with minimal label $\mathrm{body}(PC_{\mathbf{i},p})$ in $\mathrm{Search}_i(p)$
        Compute $\pi(PC_{i,p})$ according to Equation (4.6)
        $PC_{i,p} := (\pi(PC_{i,p}) :: p \leftarrow (PC_{\mathbf{i},p}))$ as in Lemma 4.2.4
        $\tilde{\mathbf{P}}_p := \tilde{\mathbf{P}}_p \cup \{PC_{i,p}\}$
        $i = i + 1$
        Compute counterfactual backbone $\pi_{i,p}$ with Equations (4.7) - (4.12) as in Lemma 4.2.5
    **end while**
    $\tilde{\mathbf{P}} := \tilde{\mathbf{P}} \cup \tilde{\mathbf{P}}_p$
**end for**
```

First, we initialize $\tilde{\mathbf{P}} := \emptyset$. Further, we choose a proposition $p \in \mathfrak{P}$ and initialize $\tilde{\mathbf{P}}_p := \emptyset$. If $\pi(p \mid \mathrm{do}(\mathbf{f})) = 0$ for all frames $\mathbf{f}$, we proceed to the next proposition $q \in \mathfrak{P} \setminus \{p\}$. Otherwise, Lemma 4.2.4 uncovers a clause $PC_{0,p} \in \mathbf{P}$ with $\mathrm{head}(PC_{0,p}) = p$, and we set $\tilde{\mathbf{P}}_p := \{PC_{0,p}\}$.

Lemma 4.2.5 computes the counterfactual backbone $\pi_{1,p}$ of the program $\mathbf{P}_{1,p} := \mathbf{P} \setminus \{PC_{0,p}\}$. If we find that $\pi_{1,p}(p \mid \mathrm{do}(\mathbf{f})) = 0$ for all frames $\mathbf{f}$ of $p \in \mathfrak{P}$, we know that $PC_{0,p} \in \mathbf{P}$ was the only clause of $\mathbf{P}$ with $\mathrm{head}(PC_{0,p}) = p$ and proceed to the next proposition $q \in \mathfrak{P} \setminus \{p\}$. Otherwise, we apply Lemma 4.2.4 to the counterfactual backbone $\pi_{1,p}$ to get a clause $PC_{1,p} \in \mathbf{P}_{1,p} := \mathbf{P} \setminus \{PC_{0,p}\}$ with $\mathrm{head}(PC_{1,p}) = p$.

We repeatedly recover clauses with head $p$ in the programs $\mathbf{P}_{i,p} := \mathbf{P} \setminus \{PC_{1,p}, ..., PC_{i,p}\}$ and set $\tilde{\mathbf{P}}_p := \{PC_{1,p}, ..., PC_{i,p}\}$ until we find $\pi_{n,p}(p \mid \mathrm{do}(\mathbf{f})) = 0$ for all frames $\mathbf{f}$, where $\pi_{n,p}$ is the counterfactual backbone of the program $\mathbf{P}_{n,p}$. If this is the case, according to Occam's razor, we conclude that $P_{n,p}$ contains no clause with head $p$ and proceed to the next proposition $q \in \mathfrak{P} \setminus \{p\}$. Finally, we obtain the ProbLog program $\tilde{\mathbf{P}} := \bigcup_{p \in \mathfrak{P}} \tilde{\mathbf{P}}_p$.

**Example 4.2.6.** Recall the situation in Example 4.2.4. Lemma 4.2.5 now computes the following counterfactual backbone of the program $\mathbf{P}_{1,wet} := \mathbf{P} \setminus \{PC_{0,wet}\}$.

$$\begin{aligned}
&\pi(\mathit{cloudy}) = 0.5,\ \pi(\mathit{sprinkler} \mid \mathrm{do}(\mathit{cloudy})) = 0.1,\ \pi(\mathit{sprinkler} \mid \mathrm{do}(\neg \mathit{cloudy})) = 0.7,\\
&\pi(\mathit{rain} \mid \mathrm{do}(\mathit{cloudy})) = 0.6,\ \pi(\mathit{rain} \mid \mathrm{do}(\neg \mathit{cloudy})) = 0,\\
&\pi(\mathit{wet} \mid \mathrm{do}(\mathit{rain}, \mathit{sprinkler})) = 0.8, ...,\ \pi(\mathit{wet} \mid \mathrm{do}(\neg \mathit{rain}, \neg \mathit{sprinkler})) = 0,\\
&\pi(\mathit{slippery} \mid \mathrm{do}(\mathit{wet})) = 0.9,\ \pi(\mathit{slippery} \mid \mathrm{do}(\neg \mathit{wet})) = 0,\\
&\pi(\mathit{sprinkler} \mid \mathit{cloudy}, \mathit{sprinkler}, \mathrm{do}(\neg \mathit{cloudy})) = 0.7,\ \pi(\mathit{rain} \mid \mathit{cloudy}, \mathit{rain}, \mathrm{do}(\neg \mathit{cloudy})) = 0,\\
&\pi(\mathit{sprinkler} \mid \neg \mathit{cloudy}, \mathit{sprinkler}, \mathrm{do}(\mathit{cloudy})) = 0.1,\ \pi(\mathit{rain} \mid \neg \mathit{cloudy}, \mathit{rain}, \mathrm{do}(\mathit{cloudy})) = 0,\\
&\pi(\mathit{wet} \mid \mathit{sprinkler}, \mathit{rain}, \mathit{wet}, \mathrm{do}(\neg \mathit{sprinkler}, \neg \mathit{rain})) = 0,\\
&\pi(\mathit{wet} \mid \mathit{sprinkler}, \mathit{rain}, \mathit{wet}, \mathrm{do}(\mathit{sprinkler}, \neg \mathit{rain})) = 0,\ ...,\\
&\pi(\mathit{wet} \mid \neg \mathit{sprinkler}, \mathit{rain}, \mathit{wet}, \mathrm{do}(\mathit{sprinkler}, \mathit{rain})) = 1,\\
&\pi(\mathit{slippery} \mid \mathit{wet}, \mathit{slippery}, \mathrm{do}(\neg \mathit{wet})) = 0,\ \pi(\mathit{slippery} \mid \neg \mathit{wet}, \mathit{slippery}, \mathrm{do}(\mathit{wet})) = 0
\end{aligned}$$

This allows us to apply Lemma 4.2.4 again to uncover the clause

$$PC_{1,wet} := (0.8 :: \mathit{wet} \leftarrow \mathit{rain}) \in \mathbf{P}_{1,wet} \subseteq \mathbf{P}.$$

In $\mathbf{P}_{2,wet} := \mathbf{P} \setminus \{PC_{0,wet}, PC_{1,wet}\}$ we find $\pi(\mathit{wet} \mid \mathrm{do}(\mathbf{f})) = 0$ for all frames $\mathbf{f}$ of $\mathit{wet}$, and we conclude that we recovered all clauses $PC \in \mathbf{P}$ with $\mathrm{head}(PC) = \mathit{wet}$.

Procedure 4.2.2 enables us to prove the following theorem, which is the main result of this chapter.

**Theorem 4.2.6** (Main Result)**.** *The counterfactual probabilities $\pi(\phi \mid \boldsymbol{e}, \mathrm{do}(\boldsymbol{i}))$ determine each acyclic proper ProbLog program $\boldsymbol{P}$ in normal form that satisfies Occam's razor up to syntactic equality. In particular, we find that Procedure 4.2.2 terminates with $\boldsymbol{P} = \tilde{\boldsymbol{P}}$.*

*Proof.* For a frame $\mathbf{f}$, we observe that $\pi(p \mid \mathrm{do}(\mathbf{f})) > 0$ only when there exists a clause $PC \in \mathbf{P}$ with $\mathrm{head}(PC) = p$. By Occam's razor, this implies that $\pi(p, \mathbf{f}) > 0$, and Proposition 4.2.2 ensures the presence of an edge in the clause search graph $\mathrm{Search}(p)$. Therefore, Lemma 4.2.4 not only guarantees the termination of Procedure 4.2.2 but also ensures that $\tilde{\mathbf{P}} \subseteq \mathbf{P}$.

Assume that there is a clause $PC \in \mathbf{P} \setminus \tilde{\mathbf{P}}$, and let $\tilde{\mathbf{P}}_p := \{PC_{1,p}, ..., PC_{n,p}\}$ be the set of all clauses in $\tilde{\mathbf{P}}$ with $\mathrm{head}(PC_{i,p}) = p = \mathrm{head}(PC)$. In particular, if $PC \in \mathbf{P} \setminus \tilde{\mathbf{P}}_p$, by properness extending $\mathrm{body}(PC)$ to a frame $\mathbf{f}$ yields $\pi_{p,n}(p \mid \mathrm{do}(\mathbf{f})) > 0$, contradicting the construction of $\tilde{\mathbf{P}}_p$ in Procedure 4.2.2. □

## 4.3 Second Result: Equivalence of Knowledge and Counterfactual Reasoning

According to Theorem 4.2.6, counterfactual reasoning uniquely determines the particular probabilistic Aristotelian knowledge expressed in acyclic proper ProbLog programs in normal form that satisfy Occam's razor. Proposition 4.1.1 interprets the assumption that a ProbLog program is in normal form as a pre-processing step, which implies that it is not a restriction on our result. Additionally, when given a set of ProbLog clauses, the assertion of properness only excludes a parameter set of Lebesgue measure zero. Hence, in some sense, almost all ProbLog programs are proper. Notably, Example 4.1.4 describes a special case from a probabilistic perspective.

Acyclicity and Occam's razor remain the primary restrictions in Theorem 4.2.6. Example 4.2.1 demonstrates that acyclicity is necessary to obtain Statement i) in Proposition 4.2.2. We conclude that we require acyclicity to establish the identities (4.1) and (4.3) in Lemma 4.2.1, on which our proof relies. However, at the current stage, we are unable to provide an example of two cyclic proper ProbLog programs in normal form that satisfy Occam's razor and yield the same counterfactual semantics. Since Occam's razor is widely used as a heuristic in machine learning, and humans tend to prefer simple explanations over complicated ones, we consider it a mild restriction of Theorem 4.2.6.

In summary, our result suggests that counterfactual reasoning determines almost all well-written ProbLog programs, offering full explanation up to syntactic equality. Here, "well-written" refers to programs in normal form that satisfy Occam's razor, and "almost all" means that for a fixed ProbLog program, the statement may only fail for a parameter set of Lebesgue measure zero. Overall, ProbLog seems to be the correct choice of language for counterfactual reasoning in the Boolean case. Furthermore, we conclude that probabilistic Aristotelian knowledge not only enables counterfactual reasoning but is often equivalent to counterfactual reasoning.

# Chapter 5

# Data-Based Counterfactual Reasoning

## Statement of Contribution

This chapter builds upon the material presented by Rückschloß and Weitkämper (53), where Kilian Rückschloß contributed the main idea and proofs of all results, while Felix Weitkämper contributed the running example.

This chapter improves on that contribution with:

- more detailed proofs;
- placing our results in the context of Theorem 3.4.6;
- a discussion of the assumptions in Theorem 5.2.1.

## 5.1 Counterfactual Reasoning and Program Induction

While observing the world, humans often draw counterfactual conclusions – reasoning about how events might have unfolded under different circumstances. In artificial intelligence, we now also aim to infer models of the world that support counterfactual reasoning based on observational data.

Kiesel's WhatIf solver (34) establishes the counterfactual reasoning of Theorem 3.4.6 for ProbLog programs (21; 26) that provide full explanations. However, note that ProbLog programs are commonly induced from observations sampled from a distribution of interest. Is it feasible to use these programs for counterfactual reasoning? Is the counterfactual reasoning these programs provide uniquely determined by the induced distribution in Definition 3.2.5?

As in Chapter 4, we fix a propositional alphabet $\mathfrak{P}$ and restrict ourselves to the study of acyclic ProbLog programs within the fragment of ProbLog clauses.

**Convention.** In the following, we restrict the term **ProbLog program** to programs in the fragment of ProbLog clauses that provide full explanation. In fact, we identify such a ProbLog program with a set $\mathbf{P}$ of ProbLog clauses.

The following example illustrates that counterfactual reasoning, as provided by ProbLog programs, is generally not uniquely determined by the underlying distribution.

**Example 5.1.1.** Assume, for example, that a patient is treated, denoted *treatment*, with probability 0.5. If we treat a patient, we expect that he recovers, denoted *recovery*, with a probability of 0.7; otherwise, he recovers with a probability of 0.5. The resulting distribution can be encoded with the following two programs $\mathbf{P}_{1/2}$.

$$\begin{array}{llll} \mathbf{P}_1: & 0.5 :: \mathit{treatment} & 0.5 :: \mathit{recovery} & 0.4 :: \mathit{recovery} \leftarrow \mathit{treatment} \\ \mathbf{P}_2: & 0.5 :: \mathit{treatment} & 0.5 :: \mathit{recovery} \leftarrow \neg \mathit{treatment} & 0.7 :: \mathit{recovery} \leftarrow \mathit{treatment} \end{array}$$

Moreover, assume that the patient recovers while not treated. What is the probability that he would have recovered under treatment?

In both programs $\mathbf{P}_{1/2}$, we conclude from our observations that the patient recovers due to the second clause, i.e., we conclude that the second clause holds in the situation we observe. If we had also treated the patient under program $\mathbf{P}_1$, he would still have recovered, since the second clause in $\mathbf{P}_1$ still applies under treatment. Hence, we obtain a probability of one for the patient to recover under treatment. Whereas, in program $\mathbf{P}_2$, the second clause is not applicable under treatment. In this case, if the patient were treated in program $\mathbf{P}_2$, he could only recover due to the third clause, resulting in a probability of 0.7 for recovery under treatment.

Moreover, the programs $\mathbf{P}_{1/2}$ give rise to the same dependence graph and predict the same effect for every intervention.

As we can see, the classical distribution semantics defined in Definition 3.2.5 does not uniquely determine the outcome of a counterfactual query. Therefore, even with perfect learning, we can only ensure that we obtain a program representing the correct distribution. In particular, each structure learning algorithm fed with samples from an observed distribution cannot distinguish between the programs $\mathbf{P}_1$ and $\mathbf{P}_2$; that is, we cannot guarantee that a learned program will accurately answer counterfactual queries.

Furthermore, as demonstrated in Chapter 4, we find that probabilistic Aristotelian knowledge not only enables counterfactual reasoning but, in many cases, the probabilistic Aristotelian knowledge represented by a ProbLog program is equivalent to the ability of counterfactual reasoning itself. From this we can conclude that inferring models that support counterfactual reasoning often involves determining ProbLog programs up to syntactic equality. In this chapter, we thus introduce a fragment of ProbLog in which each program is determined up to syntactic equality by its dependence graph, i.e., the causal diagram and the corresponding distribution. Furthermore, we argue that this framework provides a suitable setting for the currently available structure learning methods described in Section 2.2.4 that supports counterfactual reasoning.

## 5.2 An Assumption for Deriving Causal Knowledge from Observations

Generally, in structure learning, the objective is to derive a program that describes a given data set based on prior knowledge encoded by a setting (as defined in Section 2.2.4). In most cases, the data consist of observations. Additionally, we assume that our data consist of samples drawn from the distribution induced by a hidden ProbLog program $\tilde{\mathbf{P}}$ of interest, and the prior knowledge includes the correct language bias – specifically, the dependence graph of $\tilde{\mathbf{P}}$.

To assess how well a candidate program $\mathbf{P}$ represents our dataset, we use statistical tests. However, these tests only measure how well the induced distribution of the program $\mathbf{P}$ fits a given set of observations. They do not provide information on the causal mechanism that generates our data. Consequently, we cannot determine whether the causal mechanism represented by a candidate ProbLog program $\mathbf{P}$ aligns with the causal mechanism underlying our data – the one described by $\tilde{\mathbf{P}}$.

**Example 5.2.1.** Consider the programs $\mathbf{P}_1$ and $\mathbf{P}_2$ of Example 5.1.1. Although they represent different causal models leading to different counterfactual estimations, they yield the same distribution semantics and share the same dependence graph.

Thus, whenever we choose $\tilde{\mathbf{P}} := \mathbf{P}_1$ or $\tilde{\mathbf{P}} := \mathbf{P}_2$ as the hidden program, we sample from the same distribution in both cases. This means that even with the correct language bias, a structure learning algorithm cannot determine which of the two programs actually generated the provided data unless additional knowledge is provided.

More drastically, Example 5.2.1 illustrates that without additional prior knowledge, even under the assumption of perfect learning, it is only possible to learn a program $\mathbf{P}$ that represents the correct distribution. Specifically, we cannot guarantee that a learned program $\mathbf{P}$ will answer counterfactual queries correctly.

Theorem 4.2.6 states that counterfactual reasoning determines acyclic proper ProbLog programs in normal form that satisfy Occam's razor up to syntactic equality. If we now once again restrict ourselves to this fragment of ProbLog and assume that the hidden ProbLog program $\tilde{\mathbf{P}}$ and the learned ProbLog program $\mathbf{P}$ are acyclic proper in normal form and satisfy Occam's razor, learning a program $\mathbf{P}$ with correct counterfactual reasoning amounts to reconstructing $\tilde{\mathbf{P}}$ up to syntactic equality. The main result of this chapter now states that all acyclic proper positive ProbLog programs in normal form that satisfy Occam's razor are uniquely determined by their dependence graph and their underlying distribution.

**Theorem 5.2.1.** *Every acyclic proper positive ProbLog program in normal form $\boldsymbol{P}$ that satisfies Occam's razor is determined up to syntactic equality by its dependence graph* $\mathrm{graph}(\boldsymbol{P})$ *and the induced distribution $\pi$.*

*Proof.* We proceed by induction on the number $n$ of nodes in the dependence graph $\mathrm{graph}(\mathbf{P})$.

$n = 1$: In this case, the program $\mathbf{P}$ consists only of one clause $\pi :: p \leftarrow$. Hence, we set $\pi := \pi(p)$ and are done.

$n > 1$: Choose a sink $h \in \mathfrak{P}$ of $\mathrm{graph}(\mathbf{P})$. Furthermore, denote by $\mathbf{P} \setminus h$ the program that results from $\mathbf{P}$ if we erase all clauses with head $h$. Due to maximality, $h$ does not occur in the body of any other clause. In other words, $\mathbf{P} \setminus h$ induces the same distribution on $\mathfrak{P} \setminus \{h\}$ as the original program $\mathbf{P}$, and it has the graph $\mathrm{graph}(\mathbf{P}) \setminus h$ as its dependence graph. Here, $\mathrm{graph}(\mathbf{P}) \setminus h$ refers to the graph resulting from $\mathrm{graph}(\mathbf{P})$ when we remove the node $h$ along with all the edges that point to it. Finally, note that the program $\mathbf{P} \setminus h$ is acyclic, proper, positive, and in normal form, satisfying Occam's razor. Therefore, by the induction hypothesis, we can reconstruct the program $\mathbf{P} \setminus h$ from the given data.

We are now left with reconstructing the clauses that define $h$ itself. Observe that the parents $b \in \mathrm{pa}(h)$ of $h$ in $\mathrm{graph}(\mathbf{P})$ are the only propositions that may appear in the body of a clause defining $h$. Furthermore, each of these occurrences is positive. We consider the function

$$\mathrm{Ind}_h^{\mathbf{P}} : \mathcal{P}(\mathrm{pa}(h)) \to [0,1] \qquad\qquad T \mapsto \pi(h | \{t, \neg s : t \in T,\ s \in \mathrm{pa}(h) \setminus T\}),$$

where $\mathcal{P}(\_)$ denotes the power set operator. Let $T \subseteq \mathrm{pa}(h)$ and assume that

$$\pi\left( h \wedge \bigwedge_{t \in T} t \wedge \bigwedge_{s \in \mathrm{pa}(h) \setminus T} \neg s \right) > 0 \tag{5.1}$$

to obtain

$$\mathrm{Ind}_h^{\mathbf{P}}(T) = \pi(h | \{t, \neg s : t \in T,\ s \in \mathrm{pa}(h) \setminus T\}) = \frac{\pi\left( h \wedge \bigwedge_{t \in T} t \wedge \bigwedge_{s \in \mathrm{pa}(h) \setminus T} \neg s \right)}{\pi\left( \bigwedge_{t \in T} t \wedge \bigwedge_{s \in \mathrm{pa}(h) \setminus T} \neg s \right)} \overset{\text{Corollary 2.2.4}}{=}$$

$$= \frac{\pi\left( \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{body}(PC) \subseteq T \\ \mathrm{head}(PC) = h}} u(PC) \wedge \bigwedge_{t \in T} t \wedge \bigwedge_{s \in \mathrm{pa}(h) \setminus T} \neg s \right)}{\pi\left( \bigwedge_{t \in T} t \wedge \bigwedge_{s \in \mathrm{pa}(h) \setminus T} \neg s \right)}. \tag{5.2}$$

As the program $\mathbf{P}$ is acyclic, Corollary 2.2.4 yields for every selection $\epsilon$ that we find

$$\omega \models \bigwedge_{t \in T} t \wedge \bigwedge_{s \in \mathrm{pa}(h) \setminus T} \neg s$$

for the stable model $\omega$ of $\mathrm{LP}(\mathbf{P}) \cup \epsilon$ if and only if

$$\epsilon \models \bigwedge_{p \in T} \left( \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{head}(PC) = p \\ \mathrm{body}(PC) \subseteq T}} u(PC) \right) \wedge \bigwedge_{\substack{PC \in \mathbf{P} \\ q \in \mathfrak{P} \setminus (T \cup \{h\}) \\ \mathrm{head}(PC) = q \\ \mathrm{body}(PC) \subseteq T}} \neg u(PC). \tag{5.3}$$

Combing Equation (5.2) and (5.3) with the independence of the error terms yields

$$\mathrm{Ind}_h^{\mathbf{P}}(T) := \pi\left( \bigvee_{\substack{PC \in \mathbf{P} \\ \mathrm{body}(PC) \subseteq T \\ \mathrm{head}(PC) = h}} u(PC) \right) = \sum_{\substack{PC_1, \dots, PC_k \in \mathbf{P} \\ k \in \mathbb{N},\ \mathrm{body}(PC_i) \subseteq T \\ \mathrm{head}(PC_i) = h}} (-1)^k \prod_{i=1}^{k} \pi(PC_i). \tag{5.4}$$

Finally, assume that $T \subseteq \mathrm{pa}(\mathbf{P})$ is the body of a clause in $\mathbf{P}$. Applying Occam's razor, we conclude that Equation (5.1) holds, and Equation (5.4) implies that $\mathrm{Ind}_h^{\mathbf{P}}(S) < \mathrm{Ind}_h^{\mathbf{P}}(T)$ for all $S \subseteq \mathrm{pa}(h)$ with $S \subsetneq T$.

On the other hand, suppose that $\mathrm{Ind}_h^{\mathbf{P}}(S) < \mathrm{Ind}_h^{\mathbf{P}}(T)$ for all $S \subseteq \mathrm{pa}(h)$ with $S \subsetneq T$. We deduce from Equation (5.4) that $T \subseteq \mathrm{pa}(\mathbf{P})$ constitutes the body of a clause $PC \in \mathbf{P}$ with head $h$. Furthermore, since $\mathbf{P}$ is in normal form, we recursively obtain the parameter $\pi(PC)$ for every clause $PC \in \mathbf{P}$: If the body of $PC$ is minimal in $\mathbf{P}$ we find $\pi(PC) = \mathrm{Ind}_h^{\mathbf{P}}(\mathrm{body}(PC))$. Additionally, during the recursion step, Equation (5.4) yields a one-dimensional linear equation for the parameter of interest.

□

**Remark 5.2.1.** If we estimate the functions $\mathrm{Ind}_h^{\mathbf{P}}$ using relative frequencies, we obtain a structure learning algorithm that recovers an acyclic, proper, positive ProbLog program in normal form, satisfying Occam's razor. This algorithm relies on a known causal diagram and a sufficiently large set of independent samples. The resulting distributions (one for every counterfactual query) are guaranteed to converge in probability.

Finally, let us assume that we apply a structure learning algorithm as described in Section 2.2.4, with a language bias that encodes the dependence graph $\mathrm{graph}(\tilde{\mathbf{P}})$, to obtain a program $\mathbf{P}$. Furthermore, assume that we have learned perfectly, that is, that the program $\mathbf{P}$ encodes the same distribution as $\tilde{\mathbf{P}}$. Now, if we also assume that both programs $\mathbf{P}$ and $\tilde{\mathbf{P}}$ are acyclic, proper, positive ProbLog programs in normal form that satisfy Occam's razor, Theorem 5.2.1 implies that $\mathbf{P}$ and $\tilde{\mathbf{P}}$ are syntactically equal. In other words, $\mathbf{P}$ captures the causal content of $\tilde{\mathbf{P}}$.

Since, without background knowledge, each currently available structure learning algorithm only searches for positive programs that fit a given dataset, the assumption to learn positive ProbLog programs in normal form is easily realized. Causally, the absence of background knowledge leads us to assume that our data is generated by a positive ProbLog program.

**Corollary 5.2.2.** *Assume that we are given data sampled from a hidden proper positive ProbLog program in normal form $\tilde{\boldsymbol{P}}$ that satisfies Occam's razor, and assume that we are aware of the dependence graph* $\mathrm{graph}(\tilde{\boldsymbol{P}})$ *of* $\tilde{\boldsymbol{P}}$.

*Every structure learning algorithm that can learn a proper positive ProbLog program in normal form that satisfies Occam's razor with the correct dependence graph and the correct distribution reconstructs $\tilde{\boldsymbol{P}}$ from the provided data up to syntactic equality. In particular, the result of this structure learning algorithm supports counterfactual reasoning.* □

## 5.3 Third Result: A Language Bias for Data-Based Counterfactual Reasoning

Example 5.1.1 in Section 5.1 illustrates that the distribution semantics of Definition 3.2.5 does not uniquely determine the results of counterfactual queries for ProbLog programs. Consequently, utilizing currently available structure learning algorithms for counterfactual reasoning is infeasible. Additionally, Example 5.1.1 highlights the insufficiency of factual knowledge, represented by a distribution and causal relations encoded in a causal diagram, for Boolean counterfactual reasoning. Interestingly, this finding contrasts with the theory of linear functional causal models (41, §5.2.1).

Furthermore, we restrict our focus to the fragment of acyclic proper ProbLog programs in normal form that satisfy Occam's razor. According to Theorem 4.2.6, inducing a program with correct counterfactual reasoning is tantamount to reconstructing the program describing the causal generating mechanism of the data up to syntactic equality. Importantly, learning programs under a coarser notion of equivalence is insufficient for counterfactual reasoning. Our main result, Theorem 5.2.1, establishes that acyclic proper positive ProbLog programs in normal form that satisfy Occam's razor are determined up to syntactic equality by their distribution and dependence graph. Consequently, when provided with the correct language bias, the currently available structure learning algorithms described in Section 2.2.4 can recover these programs, thus supporting counterfactual reasoning in this setting.

As discussed in Section 4.3, the restriction of a ProbLog program to be proper in normal form can be considered mild. Occam's razor is also a reasonable and common assumption in program induction. Therefore, acyclicity and positivity remain the primary restrictions in Theorem 5.2.1.

The necessity of the positivity assumption is illustrated in Example 5.1.1. Additionally, acyclicity is crucial for obtaining Equation (5.4), which underpins the proof of Theorem 5.2.1. Exploring the removal of the acyclicity assumption from Theorem 5.2.1 could be an interesting direction for future research. Furthermore, determining the exact equivalence classes of ProbLog programs representing the same distributions and predicting the behavior of available structure learning algorithms for more general fragments of ProbLog are further promising directions for future work.

# Chapter 6

# Perspectives for Further Research and Conclusion

In this concluding chapter, we embed the material presented so far in a broader context and outline directions for future research. We begin by recalling three core concepts for reasoning in artificial intelligence, namely uncertainty, relations among individuals, and causality, along with their mathematical formalization.

## State of the Art: Three Core Concepts of Reasoning

When contemplating the world around us, we often encounter uncertainty in our knowledge. For example, when planning a vacation, we may be uncertain about the weather at our destination. Although we might assess that rainy weather in South Tyrol is unlikely during September, we cannot be entirely sure it will not rain. This thesis shows how to use probabilities to represent uncertainty about knowledge, an approach that results, among others, in probabilistic graphical models such as Bayesian networks and Markov random fields (42, Chapter 3).

In organizing the real world, we rely on relationships between components or elements of a given domain. Take, for instance, a wedding party, where our reasoning involves the domain of guests. Certain guests may be friends with each other, children of another guest, or married to each other. These relationships play a crucial role in structuring our understanding of the world. In this thesis, we employ first-order logic to formalize relations between individuals of a given domain. This logic-based approach leads us to the framework of logic programming in Section 2.2.4.

Combining probabilistic and logic-based approaches, the field of statistical relational artificial intelligence (44) aims to reason about uncertain knowledge regarding relations between elements of a given domain. Returning to the example of the wedding party, we may assume that friends of smokers are more likely to smoke themselves, meaning that we are uncertain about whether friends of smokers necessarily smoke. In addition, we may assume that all individuals with the same number of smoking friends are equally likely to smoke themselves. Such interchangeability assumptions may speed up calculations (11) or reduce the space of possible descriptions for a dataset (28) in the learning task. Statistical relational artificial intelligence gives rise, among other things, to Markov logic networks (46) and probabilistic logic programming under the distribution semantics (55; 43), which we discuss in Sections 2.2.3 and 2.2.4.

We often explain the world in terms of causal mechanisms. We typically understand our environment by explaining observations, i.e., effects, through self-evident prior knowledge, i.e., causes. Consider the scenario of a car accident in which a vehicle collides with a pedestrian. If we consider the vehicle hitting the pedestrian as self-evident prior knowledge, this provides a causal explanation for the injuries sustained by the pedestrian. This leads us to the conclusion: “The collision with the vehicle caused injuries to the pedestrian.”. Therefore, we not only gain factual knowledge of the pedestrian's injuries, but we also have an explanation for these injuries resulting in Aristotelian knowledge as introduced in Section 2.1.

Pearl (41) argues that causal reasoning enables us to predict the effect of external interventions and to reason counterfactually. In our example, issuing a speed limit would constitute an external intervention that can avoid accidents caused by the driver speeding. Such considerations are relevant in domains like medicine, where we may ask what happens if we actively intervene and give the patient some medication.

Counterfactual reasoning is reasoning about "What if ...?". In our example, we may ask: "What if there had been a speed limit? Would the pedestrian still be injured?". Once we explain the observed accident with the driver speeding, we may conclude that the pedestrian would not have been injured had there been a speed limit. If we, in contrast, explain the accident with the driver being drunk, we are likely to judge that the pedestrian would still be injured had there been a speed limit. Counterfactual reasoning is relevant in fields such as pharmacy, where we may inquire: "What if a patient in the control group of a pharmaceutical study had received treatment? Would he have recovered?". In decision-making, we might ask: "What if I had taken a bus earlier? Would I still have been late for my job interview?". Counterfactual reasoning is crucial in formally defining ethical notions such as responsibility, blame, fairness, or harm (30; 7; 14). In our example, we are more likely to blame the driver if we believe that the accident had not happened if he had not been speeding. In general, counterfactuals enable humans to make sense of the past, plan future courses of action, make emotional and social judgments, and adapt their behavior (31).

This thesis essentially builds on Pearl's causal models (41) as introduced in Section 2.3.1, where he represents causal mechanisms with deterministic functional relationships. Specifically, he first distinguishes between internal, i.e., explained variables $\mathbf{V}$, and external, i.e., unexplained variables $\mathbf{U}$. The causal relations between the internal variables $V \in \mathbf{V}$ are then modeled by setting $V := f_V(\mathbf{U}, \mathbf{V})$ for deterministic functions $f_V(\mathbf{U}, \mathbf{V})$ in the internal and external variables. Solving the corresponding system of equations then yields Aristotelian knowledge, that is, we gain factual knowledge represented by the solution and an explanation in terms of the external variables. Finally, Pearl (41) introduces probability in his causal models by specifying a distribution on the external variables, resulting in probabilistic causal models. An external intervention, forcing an internal variable $V$ to attain the value $v$, is realized by replacing the equation $V := f_V(\mathbf{U}, \mathbf{V})$ with $V := v$. Counterfactual reasoning is then realized by combining interventions with observations, that is, conditional probabilities.

We use Bochman's logic of causality (10) as introduced in Section 2.3.3 to further investigate the assumption that causal mechanisms should be modeled by deterministic functional relationships. Bochman (10) aims for a logical theory of causality. Initially, he represents factual knowledge or observations about the world using propositional formulas $\phi$. Inspired by Aristotle's *Posterior Analytics* (4), he views causal reasoning as a binary relation $\Rightarrow$ on factual knowledge, i.e. propositional formulas that adhere to well-motivated axioms. Here, $\phi \Rightarrow \psi$ means that a causal explanation of $\phi$ also explains $\psi$. Representing causal reasoning, i.e., the relation $\Rightarrow$, through rules of the form $\phi \Rightarrow \psi$ for propositional formulas $\phi$ and $\psi$, results in Bochman's causal theories (10). If no cyclic causal relationships are at play, Bochman's causal theories (10) align with Pearl's causal models (41) and abductive logic programming under the supported model semantics (22). This yields a causal interpretation for acyclic abductive logic programs and a deeper understanding of the reasoning in Pearl's Boolean causal models (41).

## First Contribution: A Theory of Boolean Cyclic Causal Reasoning

This thesis focuses on causal reasoning in the Boolean case, where all variables possibly take on two values, represented by true and false. Although Pearl's causal models (41), abductive logic programs (22) and Bochman's causal theories (10) provide a good description of acyclic causal reasoning, in applications, one may also encounter situations with cyclic causal relationships. For example, in economics, an increase in price usually causes a decrease in demand, while an increase in demand causes an increase in price.

Once cyclic causal relationships are at play, Pearl's causal models (41) may not be well-defined, and on the logic programming side, we have to consider different proposals for the semantics of an abductive logic program: the supported model semantics (25), the minimal model semantics (59), and the stable model semantics (27).

In Section 3.1, we observe that Bochman's causal theories may yield counterintuitive results when cyclic causal relationships are considered. We explain these counterintuitive results by Bochman (10) allowing for cyclic and infinite explanations, and propose avoiding these cyclic and infinite explanations by explicitly stating the prior knowledge with which an explanation must start. Extending Bochman's framework (10) with this prior knowledge yields abductive causal theories, which are essentially equivalent to abductive logic programs under the stable model semantics (22) as shown in Theorem 3.1.7. We conclude that abductive logic programming (22) under the stable model semantics (27) emerges as the framework of choice for deterministic causal reasoning.

## Second Contribution: Unifying Causal Reasoning across Widespread Formalisms in Statistical Relational Artificial Intelligence

Although frameworks in statistical relational artificial intelligence often lack the dimension of causal reasoning, Pearl's causal models (41) and Bochman's causal theories (10) cannot represent relations between individuals in a given domain. Additionally, Bochman's logical theory of causality cannot handle uncertain knowledge. This thesis aims to augment statistical relational artificial intelligence with causal reasoning and facilitate probabilistic causal reasoning that captures relations in a given domain. Here, we focus on the Boolean case, where all random variables of interest take on two values, represented by true or false. To avoid reinventing the wheel for causal reasoning within every formalism in statistical relational artificial intelligence, we take a generic approach; that is, we introduce causal reasoning in a formalism that subsumes as many frameworks as possible.

Section 3.2 merges Markov logic networks (46) with abductive logic programming (22) into weighted abductive logic programming – a formalism tailored to causal reasoning under uncertainty. In Section 3.4, we demonstrate that weighted abductive logic programming encompasses several widely used formalisms in statistical relational artificial intelligence, including Markov logic networks (46), logic programs with annotated disjunctions (62), ProbLog programs (21; 26), and $\text{LP}^{\text{MLN}}$ programs (35). By embedding these formalisms into weighted abductive logic programming and transferring our causal reasoning there, we achieve consistent causal reasoning across various frameworks in statistical relational artificial intelligence. In particular, for logic programs with annotated disjunctions (62), we recover the causal reasoning provided by CP-logic (60; 61), as introduced in Section 2.3.2.

### First Research Direction: Applying Lifted Inference for Causal Reasoning

While we introduce weighted abductive logic programming as a formalism capturing interchangeability assumptions regarding the elements of a given domain, we have only considered causal reasoning independently of these interchangeability assumptions. In the field of lifted inference, a variety of algorithms have been developed that make use of such interchangeability assumptions to speed up computations of noncausal (conditional) probabilities (11). Therefore, we propose to investigate lifted inference for causal reasoning in weighted abductive logic programs, i.e., to use the interchangeability assumption encoded in a non-ground program to speed up computations of post-interventional and counterfactual probabilities as well.

### Second Research Direction: A Theory of Statistical Relational Artificial Intelligence

In Section 3.2, we construct weighted abductive logic programs such that the corresponding causal reasoning satisfies Bochman's axiomatization (10) together with the assertion that a causal explanation has to start with previously specified prior knowledge. Inspired by the maximum entropy principle, we introduce uncertainty based on the parameterization of probability spaces provided by Berger et al. (8). Finally, we commit to the principle of indifference for propagating probabilities from prior knowledge to the corresponding Aristotelian knowledge. However, so far, our approach misses a completeness result regarding these choices.

We suggest possibly generalizing weighted abductive logic programming to a formalism encompassing any framework modeling the principles that guided the construction of weighted abductive logic programs. In this way, our aim is to classify all frameworks in statistical relational artificial intelligence. In particular, we conjecture that all these frameworks should be embedded into such a general formalism. If one, nevertheless, finds a framework that cannot be embedded into such a formalism, this would raise the question of which fundamental concept is missing in the description of statistical relational artificial intelligence. Furthermore, proving that a particular object is not representable in such a general formalism, one immediately obtains that this object is not representable within every subframework of this formalism. In this way, our approach aims to facilitate nonexpressivity results in statistical relational artificial intelligence.

## Third Contribution: Probabilistic Boolean Counterfactual Reasoning

From now on, we will adopt the view that weighted abductive logic programming is the most general formalism for Boolean causal reasoning under uncertainty. Section 3.3 is devoted to counterfactual reasoning within weighted abductive logic programming. Hereby, we explicitly assert that counterfactual reasoning should consistently generalize conditional probabilities.

In Section 3.4.3, we further assert Reichenbach's common cause assumption that probabilistic dependence stems from causal reasoning. It then turns out that the fragment of weighted abductive logic programming enabling well-defined counterfactual reasoning is exactly given by ProbLog programs that provide full explanations. Furthermore, our approach to counterfactual reasoning is consistent with the work of Kiesel et al. (34). Specifically, we find that the counterfactual reasoning in ProbLog programs providing full explanation is implemented in Kiesel's WhatIf solver, extending his aspmc solver by the ability of counterfactual reasoning.

Theorem 4.2.6 proves, under rather mild assumptions, that counterfactual reasoning determines an acyclic ProbLog program up to syntactic equality. Specifically, we provide a procedure that reconstructs such a ProbLog program from its answers to all counterfactual queries. We conclude that ProbLog is a suitable formalism for counterfactual reasoning, as many ProbLog programs are in one-to-one correspondence with the desired query answers.

Suppose further that we are given a set of data sampled from the distribution of a hidden acyclic ProbLog program $\widetilde{\mathbf{P}}$. In causal structure discovery, the aim is to determine another ProbLog program $\mathbf{P}$ that describes this set of observations while providing the corresponding counterfactual reasoning of the program $\widetilde{\mathbf{P}}$. According to our result, in many cases, Theorem 4.2.6 implies that we need to reconstruct the hidden ProbLog program $\widetilde{\mathbf{P}}$ from the provided data up to syntactic equality, i.e., we need to ensure $\mathbf{P} = \widetilde{\mathbf{P}}$. Hence, causal structure discovery requires great precision in the learning process.

### First Research Direction: Concepts Based on Counterfactual Reasoning

Our work demonstrates that ProbLog programs offer a convenient way to represent causal mechanisms. Recall that the corresponding counterfactual reasoning is implemented in Kiesel's WhatIf solver and that ethical notions such as responsibility, blame, fairness, or harm are essentially formulated in counterfactuals (30; 7; 14). Hence, we suggest using ProbLog for reasoning about these ethical notions.

In this work, we focus on type causality, that is, statements like "Smoking causes cancer," and do not address individual or actual causality, that is, statements like "Uncle Sam died young because he used to smoke one pack of cigarettes a day." Note that formalizing actual causality essentially relies on counterfactual reasoning (30). Therefore, we suggest investigating actual causality within ProbLog. According to Theorem 4.2.6, in many cases, counterfactual reasoning uniquely determines our representation for type causality (ProbLog programs) up to syntactic equality. Translating this result into a statement about the relationship between actual and type causality would be particularly interesting.

### Second Research Direction: Counterfactual Reasoning in ProbLog

Recall Theorem 4.2.6, which essentially states that counterfactual reasoning determines an acyclic ProbLog program up to syntactic equality.

We only require the acyclicity assumption because our proof relies on Clark completion (25). Therefore, we propose extending our result to cyclic ProbLog programs by replacing Clark completion (25) in our proof with the more elaborate translation of logic programs provided by Lin and Zhao (38).

We also emphasize that our proof provides a procedure for recovering a given ProbLog program from the answers to the corresponding counterfactual queries. This opens the door to deriving ProbLog programs that generalize counterfactual estimations obtained, for instance, from randomized controlled trials. However, in the current stage, our procedure requires numerous counterfactual estimates to recover a ProbLog program. Hence, further research is needed to reduce the amount of counterfactual information necessary for this reconstruction process. In particular, leveraging the interchangeability assumptions encoded in non-ground ProbLog programs may be a promising starting point.

Finally, suppose that we are provided with a probability for each possible counterfactual query. In this situation, we suggest further investigating under which assumptions the provided procedure terminates with a resulting ProbLog program $\mathbf{P}$ and under which assumptions the program $\mathbf{P}$ yields the given counterfactual probabilities. This investigation would allow us to explore the counterfactual expressiveness of ProbLog, enabling the identification of examples where custom counterfactual conclusions may not be represented using ProbLog programs.

## Fourth Contribution: Relation between Counterfactual Reasoning and Cause-Effect Relationships

In Theorem 5.2.1, we employ the same mild assumptions as in Theorem 4.2.6 to prove that positive acyclic ProbLog programs are essentially determined up to syntactic equality by the corresponding distribution and cause-effect relationships. In particular, we provide a procedure that recovers a positive acyclic ProbLog program from its induced distribution and these cause-effect relationships.

Chapter 5 further demonstrates that this result fails if we allow for negation in the body of ProbLog clauses. This contrasts with the case of structural equation models, that is, causal models, which only mention linear equations, which are uniquely determined by the induced distribution and the corresponding cause-effect relationship (41, Section 5.2.1).

Suppose that we are given a set of observational data sampled from the distribution induced by a hidden positive acyclic ProbLog program $\tilde{\mathbf{P}}$. Furthermore, assume we have a structure learning algorithm that is able to learn a positive acyclic ProbLog program $\mathbf{P}$ with the correct distribution from the provided data and cause-effect relationships. In this case, this algorithm recovers the program $\tilde{\mathbf{P}}$ up to syntactic equality, i.e., we find $\mathbf{P} = \tilde{\mathbf{P}}$. In particular, it is feasible to use the resulting program $\mathbf{P}$ for counterfactual reasoning. Lastly, we argue that this observation provides a setting for the currently available structure learning algorithms in probabilistic logic programming, allowing them to determine programs supporting counterfactual reasoning.

### Research Direction: Applications of Data-Based Counterfactual Reasoning

In Chapter 5, we justify using currently available structure learning algorithms in probabilistic logic programming to derive programs that support counterfactual reasoning from observational data. As a next step, we propose applying our theory and considering probabilistic logic programming for data-based counterfactual reasoning in applications.

## Summary of Our Contributions and Proposals for Future Research

This thesis first introduces weighted abductive logic programming as a general formalism tailored to causal reasoning in Section 3.2. This formalism encompasses widespread frameworks of statistical relational artificial intelligence, facilitating consistent causal reasoning across these frameworks.

Within weighted abductive logic programming, this thesis then determines ProbLog programs that provide full explanation as a fragment enabling well-defined counterfactual reasoning in Section 3.2. Furthermore, it focuses on acyclic ProbLog programs in Chapters 4 and 5. Chapter 4 proves that counterfactual reasoning essentially determines these programs up to syntactic equality. Lastly, Chapter 5 shows that positive acyclic ProbLog programs are essentially determined up to syntactic equality by their induced distribution and cause-effect relationships.

Based on the results of this thesis, we propose the following directions for future research:

- Classifying frameworks in statistical relational artificial intelligence to enable non-expressivity results
- Developing lifted inference algorithms for causal reasoning
- Reasoning on actual causality, responsibility, blame, fairness, and harm within ProbLog
- Clarifying the relation between actual causality and type causality within ProbLog
- Learning ProbLog programs from counterfactual estimations (randomized controlled trials)
- Characterizing the counterfactual expressiveness of ProbLog
- Conducting data-based counterfactual reasoning in applications

# Bibliography

[1] Angioni, L.: Causality and Coextensiveness in Aristotle's Posterior Analytics 1. 13. In: Oxford Studies in Ancient Philosophy, Volume 54. Oxford University Press (2018), https://doi.org/10.1093/oso/9780198825128.003.0005

[2] Aquinas, T.: Summa Contra Gentiles: Book Two: Creation, Translated by Anderson, J. F. University of Notre Dame Press (1976), https://undpress.nd.edu/9780268016807/summa-contra-gentiles/

[3] Aristotle: Prior Analytics, Translated by Jenkinson, A. J. In: Complete Works of Aristotle, Volume 1: The Revised Oxford Translation. Princeton University Press (1984), https://classics.mit.edu/Aristotle/prior.html

[4] Aristotle: Posterior Analytics, Translated by Barnes, J., pp. 114–166. Princeton University Press (1985), https://doi.org/doi:10.1515/9781400835843-007

[5] Balke, A., Pearl, J.: Probabilistic evaluation of counterfactual queries. In: Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence (AAAI 1994). pp. 230–237. AAAI Press (1994), https://dl.acm.org/doi/10.5555/199288.178004

[6] Baral, C., Gelfond, M., Rushton, N.: Probabilistic reasoning with answer sets. Theory and Practice of Logic Programming (1), 57–144 (2009), https://doi.org/10.1017/S1471068408003645

[7] Beckers, S., Chockler, H., Halpern, J.Y.: A causal analysis of harm. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. Curran Associates Inc. (2024)

[8] Berger, A.L., Pietra, V.J.D., Pietra, S.A.D.: A maximum entropy approach to natural language processing. Computational Linguistics (1), 39–71 (1996), https://dl.acm.org/doi/10.5555/234285.234289

[9] Bochman, A.: Explanatory Nonmonotonic Reasoning. World Scientific (2005), https://doi.org/10.1142/5707

[10] Bochman, A.: A Logical Theory of Causality. The MIT Press (2021), https://doi.org/10.7551/mitpress/12387.001.0001

[11] Van den Broeck, G., Kersting, K., Natarajan, S., Poole, D.: An Introduction to Lifted Probabilistic Inference. MIT Press (2021), https://doi.org/10.7551/mitpress/10548.001.0001

[12] Bry, F., Eisinger, N., Eiter, T., Furche, T., Gottlob, G., Ley, C., Linse, B., Pichler, R., Wei, F.: Foundations of Rule-Based Query Answering, pp. 1–153. Springer Berlin Heidelberg (2007), https://doi.org/10.1007/978-3-540-74615-7_1

[13] Burnyeat, M.F.: Aristotle on understanding knowledge. In: Explorations in Ancient and Modern Philosophy. p. 115–144. Cambridge University Press (2012), https://doi.org/10.1017/CBO9780511974069

[14] Chockler, H., Halpern, J.Y.: On testing for discrimination using causal models. Proceedings of the AAAI Conference on Artificial Intelligence (5), 5548–5555 (2022), https://doi.org/10.1609/aaai.v36i5.20494

[15] Chomicki, J., Saake, G. (eds.): Logics for databases and information systems. Kluwer Academic Publishers, USA (1998)

[16] Clark, K.L.: Negation as failure. In: Logic and Data Bases. pp. 293–322. Springer US, Boston, MA (1978), https://dl.acm.org/doi/book/10.5555/578615

[17] Cozman, F.G., Mauá, D.D.: On the semantics and complexity of probabilistic logic programs. Journal of Artificial Intelligence Research (1), 221–262 (2017), https://dl.acm.org/doi/10.5555/3207692.3207698

[18] Cozman, F.G., Mauá, D.D.: The structure and complexity of credal semantics. In: Proceedings of the 3rd International Workshop on Probabilistic Logic Programming (2016), https://ceur-ws.org/Vol-1661/paper-01.pdf

[19] De Martino, A., De Martino, D.: An introduction to the maximum entropy approach and its application to inference problems in biology. Heliyon (4) (2018), https://doi.org/10.1016/j.heliyon.2018.e00596

[20] De Raedt, L., Demoen, B., Fierens, D., Gutmann, B., Janssens, G., Kimmig, A., Landwehr, N., Mantadelis, T., Meert, W., Rocha, R., Costa, V., Thon, I., Vennekens, J.: Towards digesting the alphabet-soup of statistical relational learning. Tech. rep., Katholieke Universiteit Leuven (2008), https://lirias.kuleuven.be/retrieve/41397

[21] De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007). pp. 2462–2467. AAAI Press (2007), https://dl.acm.org/doi/10.5555/1625275.1625673

[22] Denecker, M., Kakas, A.C.: Abduction in logic programming. In: Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I. pp. 402–436. Springer (2002), https://doi.org/10.1007/3-540-45628-7_16

[23] Duignan, B.: Occam's razor. Encyclopedia Britannica (2024), https://www.britannica.com/topic/Occams-razor

[24] Ebbinghaus, H.D., Flum, J.: Finite model theory. Springer (1995), https://doi.org/10.1007/3-540-28788-4

[25] Fages, F.: Consistency of Clark's completion and existence of stable models. Methods of Logic in Computer Science (1), 51–60 (1994), https://www.researchgate.net/publication/220492237_Consistency_of_Clark%27s_completion_and_existence_of_stable_models

[26] Fierens, D., van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., Janssens, G., De Raedt, L.: Inference and learning in probabilistic logic programs using weighted boolean formulas. Theory and Practice of Logic Programming (3), 358–401 (2015), https://doi.org/10.1017/S1471068414000076

[27] Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of International Logic Programming Conference and Symposium. pp. 1070–1080. MIT Press (1988), http://www.cs.utexas.edu/users/ai-lab?gel88

[28] Getoor, L., Taskar, B.: Introduction to statistical relational learning. The MIT Press (2007), https://doi.org/10.7551/mitpress/7432.001.0001

[29] Goldberger, A.S.: On block-recursive linear regression equations. Brazilian Journal of Probability and Statistics (1), 46–48 (1992), http://www.jstor.org/stable/43601444

[30] Halpern, J.Y.: Actual Causality. MIT Press (2016), https://doi.org/10.7551/mitpress/10809.001.0001

[31] Hoeck, N.V.: Cognitive neuroscience of human counterfactual reasoning. Frontiers in Human Neuroscience (2015), https://doi.org/10.3389/fnhum.2015.00420

[32] Hulswit, M.: Some key moments in the history of the concept of causation. In: From Cause to Causation: A Peircean Perspective. pp. 1–45. Springer Netherlands, Dordrecht (2002)

[33] Kant, I.: Critique of Pure Reason. Cambridge University Press, Cambridge (1998), https://doi.org/10.1017/CBO9780511804649

[34] Kiesel, R., Rückschloß, K., Weitkämper, F.: "What if?" in probabilistic logic programming. Theory and Practice of Logic Programming (4), 884–899 (2023), https://doi.org/10.1017/S1471068423000133

[35] Lee, J., Wang, Y.: Weighted rules under the stable model semantics. In: Proceedings of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2016). p. 145–154. AAAI Press (2016), https://dl.acm.org/doi/abs/10.5555/3032027.3032045

[36] Leibniz, G.W.: First truths. In: Philosophical Papers and Letters. pp. 267–271. Springer Netherlands (1989), https://doi.org/10.1007/978-94-010-1426-7_31

[37] Lewis, D.K.: Counterfactuals. Blackwell (1973)

[38] Lin, F., Zhao, Y.: Assat: computing answer sets of a logic program by sat solvers. Artificial Intelligence (1), 115–137 (2004), https://doi.org/10.1016/j.artint.2004.04.004

[39] Malink, M.: Aristotle on circular proof. Phronesis (3), 215–248 (2013), https://doi.org/10.1163/15685284-12341249

[40] Papantonis, I., Belle, V.: Interventions and counterfactuals in tractable probabilistic models: Limitations of contemporary transformations. arXiv.org e-Print Archive: Computing Research Repository (CoRR) (2020), https://doi.org/10.48550/arXiv.2001.10905

[41] Pearl, J.: Causality. Cambridge University Press, 2 edn. (2000), https://doi.org/10.1017/CBO9780511803161

[42] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc. (1988)

[43] Poole, D.: Probabilistic horn abduction and Bayesian networks. Artificial Intelligence (1), 81–129 (1993), https://doi.org/10.1016/0004-3702(93)90061-F

[44] Raedt, L.D., Kersting, K., Natarajan, S.: Statistical Relational Artificial Intelligence: Logic, Probability, and Computation. Morgan & Claypool Publishers (2016), https://dl.acm.org/doi/10.5555/3027718

[45] Reichenbach, H.: The Direction of Time. Dover Publications (1956)

[46] Richardson, M., Domingos, P.: Markov logic networks. Machine Learning (1–2), 107–136 (2006), https://doi.org/10.1007/s10994-006-5833-1

[47] Riguzzi, F.: Foundations of Probabilistic Logic Programming: Languages, Semantics, Inference and Learning. River Publishers (2020), https://doi.org/10.1201/9781003338192

[48] Riguzzi, F., Cota, G., Bellodi, E., Zese, R.: Causal inference in cplint. International Journal of Approximate Reasoning (2017), https://doi.org/10.1016/j.ijar.2017.09.007

[49] Rückschloß, K., Weitkämper, F.: Exploiting the full power of Pearl's causality in probabilistic logic programming. In: Proceedings of the International Conference on Logic Programming 2022 Workshops co-located with the 38th International Conference on Logic Programming (ICLP 2022). CEUR-WS.org (2022), http://ceur-ws.org/Vol-3193/paper1PLP.pdf

[50] Rückschloß, K., Weitkämper, F.: On the subtlety of causal reasoning in probabilistic logic programming: A bug report about the causal interpretation of annotated disjunctions. In: Proceedings of the International Conference on Logic Programming 2023 Workshops co-located with the 39th International Conference on Logic Programming (ICLP 2023). CEUR-WS.org (2023), https://ceur-ws.org/Vol-3437/short4PLP.pdf

[51] Rückschloß, K., Weitkämper, F.: What do counterfactuals say about the world? Reconstructing probabilistic logic programs from answers to "What if?" queries. In: Inductive Logic Programming, 32nd International Conference (ILP 2023). pp. 93–108. Springer Nature Switzerland (2023), https://doi.org/10.1007/978-3-031-49299-0_7

[52] Rückschloß, K., Weitkämper, F.: A logic for Boolean causal reasoning under uncertainty or a journey into reinventing probabilistic logic programming, https://www.plai.ifi.lmu.de/download/weitkaemper-rueckschloss/ecai_weighted_abductive_logic_programs-3.pdf, manuscript under review

[53] Rückschloß, K., Weitkämper, F.: "Would life be more interesting if I were in AI?" Answering counterfactuals based on probabilistic inductive logic programming. In: Proceedings of the 39th International Conference on Logic Programming (ICLP 2023). pp. 58–64. Open Publishing Association (2023), https://dx.doi.org/10.4204/EPTCS.385.7

[54] Sandstad, P.: The formal cause in the posterior analytics. Filozofski Vestnik (3), 7–26 (2016), https://centaur.reading.ac.uk/100246/

[55] Sato, T.: A statistical learning method for logic programs with distribution semantics. In: Logic Programming, Proceedings of the Twelfth International Conference on Logic Programming. pp. 715–729. MIT Press (1995)

[56] Shafer, G.: The Art of Causal Conjecture. MIT Press (1996), https://doi.org/10.7551/mitpress/1403.001.0001

[57] Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal (3), 379–423 (1948). https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[58] Tarski, A.: The concept of truth in formalized languages, translated by Woodger, J. H. In: Logic, semantics, metamathematics, pp. 152–278. Clarendon Press (1956)

[59] Van Emden, M.H., Kowalski, R.A.: The semantics of predicate logic as a programming language. Journal of the Association for Computing Machinery (4), 733–742 (1976), https://doi.org/10.1145/321978.321991

[60] Vennekens, J., Denecker, M., Bruynooghe, M.: CP-logic: A language of causal probabilistic events and its relation to logic programming. Theory and Practice of Logic Programming (3), 245–308 (2009), https://doi.org/10.1017/S1471068409003767

[61] Vennekens, J., Bruynooghe, M., Denecker, M.: Embracing events in causal modelling: Interventions and counterfactuals in CP-logic. In: Logics in Artificial Intelligence, Twelfth European Conference, (JELIA 2010). pp. 313–325. Springer Berlin Heidelberg (2010), https://doi.org/10.1007/978-3-642-15675-5_27

[62] Vennekens, J., Verbaeten, S., Bruynooghe, M.: Logic programs with annotated disjunctions. In: Logic Programming, 20th International Conference (ICLP 2004). pp. 431–445. Springer Berlin Heidelberg (2004), https://doi.org/10.1007/978-3-540-27775-0_30

[63] Williamson, J.: Philosophies of probability. In: Philosophy of Mathematics, pp. 493–533. North-Holland (2009), https://doi.org/10.1016/B978-0-444-51555-1.50016-X