UNIVERSITY COLLEGE LONDON (UCL)

Probabilistic Epistemic Reasoning about Actions

PHD THESIS

Fabio Aurelio D'Asaro

Declaration

I, Fabio Aurelio D'Asaro, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Modelling agents that are able to reason about actions in an ever-changing environment continues to be a central challenge in Artificial Intelligence, and many technical frameworks that tackle it have been proposed over the past few decades. This thesis deals with this problem in the case in which the environment and its evolution is incompletely known, and agents can seek to gain further information about it and act accordingly. Two languages are proposed, namely PEC+ and EPEC, which extend a standard logical language for reasoning about actions known as the Event Calculus, and use Probability Theory as a measure of the agent's degree of belief about aspects of the domain. These languages are then shown to satisfy some essential properties. PEC+ is implemented and tested against a number of real world scenarios.

Impact Statement

Artificial Intelligence (AI) is a branch of computing that nowadays has central importance in both the industrial and academic world. This thesis presents ideas and methods belonging to this highly impactful area, and this research has been undertaken with the expectation that it will eventually be applied to realise applications for automated tasks such as automated forecasting, diagnosis and explanation in highly noisy environments. Some suggestions for possible applications are presented in the thesis, and include e.g. applications to medical reasoning. The frameworks presented here already constitute a further step towards a full integration between two fields relevant to AI, namely Reasoning About Actions and Probability Theory, and a successful merger between these two areas is widely recognised to be of key importance in furthering the field and impact of AI research.

Acknowledgements

Being supervised by Antonis Bikakis, Luke Dickens and Rob Miller has been a true privilege. To them I owe many of the ideas presented here, as well as a lot of good fun and humour.

I also wish to thank my family for their support throughout these years, and to all of my friends (you know who you are!) in London, Manchester and Palermo.

Contents

Chapter 1

Introduction

Endowing computing machines with human-like intelligence has been one of the major long-term goals of Computer Science since its early days. A. M. Turing, in one of the most influential philosophical papers of the computing era [65], wrote in a concluding remark: "We may hope that machines will eventually compete with men in all purely intellectual fields", and then asked "But which are the best ones to start with? Even this is a difficult decision". Since then, researchers in *Artificial Intelligence* (*AI* for short) have focused on a variety of intellectual tasks, successfully tackling many of them and even managing to outperform humans at some. However, human beings still remain unrivalled on a range of activities including (but not at all limited to) natural language comprehension and processing, vision, decision-making and planning in unconstrained environments. Some of these tasks are commonly considered to be so hard that even their sub-problems are sometimes thought to be equivalent to the very problem of intelligence, as exemplified by D. Hofstadter in the context of natural language processing:

> It is amazing how deep this problem with the word "the" is. It is probably safe to say that writing a program which can fully handle the top five words of English – "the", "of", "and", "a", and "to" – would be equivalent to solving the entire problem of AI, and hence tantamount to knowing what intelligence and consciousness are. [32]

Problems that are as hard as the problem of AI itself have been recently dubbed *AI-Complete*[1] and, arguably, include the problem of reasoning about *actions* occurring along a time dimension and their interaction with the environment – which is the focus of the present work. That this problem is computationally complex is testified for example by the fact that it has driven researchers into investigating a number of non-trivial sub-problems stemming from it – most notably the *Frame Problem*, the *Qualification Problem* and the *Ramification Problem* discussed below. Furthermore, it has drawn attention even from neighbouring fields such as Cognitive Science and Philosophy (see e.g. [15] and [59] for a survey). Finally, it is linked to other central issues in AI: in Linguistics, methods for dealing with it have been successfully applied e.g. to the semantics of tenses [66], which is relevant to the problem of natural language processing discussed above.

[1] By analogy to "NP-Complete" in complexity theory, see e.g. `https://en.wikipedia.org/wiki/AI-complete`, accessed 1 Dec 2017.

These introductory remarks serve to pinpoint the importance of the problem of Reasoning about Actions in the context of AI, and to highlight its open-ended nature. Indeed, although some technical solutions are already available, there is still a long way to go before human performance is matched or even before the problem is satisfactorily formalised and a consensus is reached about the form of its solution. This constitutes a first motivation for this thesis, which proposes the integration of logical machinery – the favoured tool in this area – with models of uncertainty (*Probability Theory* in particular) and epistemic features, hence changing the shape of the usual (logic-based) solution. The types of reasoning that are relevant to this thesis are informally introduced in section 1.1. Finally, it is important to note that efforts in this direction have had and will continue to have significant impact on applications as discussed in section 1.2, providing further important motivation for this work.

1.1 Epistemic Reasoning with Logic and Probability

As briefly discussed above, this thesis has its roots in the field of *Reasoning About Actions* (*RAA* for short) which deals with the problem of formally representing actions and their effects in a changing environment, in order to automatically reason about the world in a commonsensical way. For instance, one might expect a RAA framework to be able to draw deductions as simple as

(S1)
If the door is opened, the alarm will be activated,
I'll open the door tomorrow.

Tomorrow the alarm will be activated. $\therefore$

or, perhaps even more straightforwardly, one might want to infer

(S2)
The walls are yellow,
I'll open the door in one hour.

The walls will still be yellow in two hours. $\therefore$

which seems a safe and rational conclusion given the two premises. Rather disappointingly, Classical Logic, in the form of propositional and predicate calculus, does not provide a good inbuilt model of cause and effect, as dealing with them always involves a set of non-trivial background hypotheses which are implicitly assumed by human beings but need to be made explicit when formally modelling a scenario. For instance, in syllogism 1.1, the following statements are tacitly assumed: i) opening the door has no effect on the colour of the walls, ii) no one is going to paint the walls in the next two hours, or, more generally, that no other relevant action is going to be executed.

The problem of determining all the non-effects of an action is widely known as *the Frame Problem*[2]. Together with its siblings, the *Ramification Problem* (concerned with determining all the implicit effects of an action) and the *Qualification Problem* (in brief, the problem of

[2]This term was used for the first time by J. McCarthy and P. J. Hayes in [47], see e.g. [58] for a survey and a comprehensive discussion.

listing all the possible conditions preventing an action to achieve its intended effect), the frame problem has stimulated the development of several techniques to solve it, or at least get round it. Many of these efforts helped inspire the development of a non-standard logic, today known as *non-monotonic logic*, which tackles problems such as representing default assumptions about the world, without losing internal consistency of the representation.

Another feature of common sense reasoning that Classical Logic is unable to easily capture its capacity to deal with various forms of uncertainty. For instance, consider the following case:

(S3)
If it is raining, then it is cloudy,
It is raining.

It is cloudy. $\therefore$

This syllogism is supported by classical propositional calculus as it an application of *Modus Ponens*. However, Classical Logic has no way to express the following, weaker, reasoning:

(S4)
If it is raining, then it is cloudy,
It is cloudy.

It is *more plausible* that it is raining. $\therefore$

which expresses a very common way of reasoning. As a further example, consider the following excerpt from a classic probability book by E. T. Jaynes:

> Suppose some dark night a policeman walks down a street, apparently deserted; but suddenly he hears a burglar alarm, looks across the street, and sees a jewellery store with a broken window. Then a person wearing a mask comes crawling out through the broken window, carrying a bag which turns out to be full of expensive jewellery. The policeman immediately concludes that this gentleman is dishonest. [34]

The policeman's conclusion, although not strictly following from the rules of deduction, seems indeed very reasonable. Jaynes argues that in order to draw such a strong conclusion, the policeman is using in fact a very weak syllogism of the form

(S5)
If A then B becomes more plausible,
B is true.

A becomes more plausible. $\therefore$

where A is the statement "the gentleman is dishonest" and B is the conjunction of all the observations made by the policeman (hearing a burglar alarm, seeing a jewellery store with a broken window, etc...).

These examples show how any flexible enough model of common sense reasoning should be able to handle aspects of uncertainty. Many AI researchers have suggested that probability can help provide a semantics to a logic of *plausible* reasoning, which is able to retain the rigour of deductive logic and augment it with the possibility to handle softer ways of reasoning such as those introduced above.

Finally, imagine that an agent A lives in the environment being modelled, and move the focus to investigating what A *knows*. For example, syllogism (S3) can be turned into the following similar reasoning:

(S6)
A *knows* that if it is raining, then it is cloudy,
A *knows* that it is raining.

A *knows* that it is cloudy. $\therefore$

Syllogism (S6) differs from syllogism (S3) in a crucial aspect: it is not sufficient for the weather to be rainy for A to know that it is cloudy: A must also *know* that it is raining. In other words, the following syllogism *is not* valid:

(S7)
A *knows* that if it is raining, then it is cloudy,
It is raining.

A *knows* that it is cloudy. $\therefore$

In order to gain access to the truth value of the proposition "It is raining", A might want e.g. to look outside the window or listen to the newspaper: i.e., A needs to perform a *sensing action*. It is frequently assumed that if some fact is known, then that fact must be true, e.g.:

(S8)
A *knows* that it is raining.

It is raining. $\therefore$

implying that the following *is* a valid reasoning:

(S9)
If it is raining, then it is cloudy,
A *knows* that it is raining.

It is cloudy. $\therefore$

which is exactly equivalent to (S3). Under this assumption sensing actions must be *perfect*, i.e. they cannot produce knowledge that does not correspond to truth.

Things are different when mixing together *Epistemic Logic* (which deals with knowledge) and plausible reasoning. Indeed, "A knows that it is plausible that it is raining" does not imply "It is raining", i.e. one can know that some fact is plausible without that fact being true. This allows for the possibility of modelling *imperfect sensing actions*, that is, actions that produce knowledge that does not necessarily correspond to truth, as opposed to perfect sensing actions discussed above.

This work explores how these forms of reasoning can be integrated in a standard RAA language known as the *Event Calculus* (*EC* for short). In EC, time-stamped actions affecting the current state of an environment can be performed by an agent. In this thesis, EC is augmented with actions whose effects are specified probabilistically and with (imperfect) sensing actions which update the state of knowledge of the agent, where the state of knowledge is of a proba-

bilistic nature.

1.2 Application Domains and Example Scenarios

In the last few years there has been an increase of interest in RAA and logic programming: many frameworks, which were initially mainly tested against *benchmark problems* such as the *Yale Shooting Problem* [31] are now being used in real-world applications. On the other hand, this thesis also deals with Probability Theory which has recently been highly successful in AI, as demonstrated by the emergence of numerous fields of Computer Science based on probabilistic reasoning such as Machine Learning, Data Mining, Pattern Recognition and Automated Control. Arguably, a robust and scalable integration between RAA and Probabilistic Reasoning would result in a benefit for the field of Reasoning about Actions, which in turn would have a positive impact on its applications. Similarly to combining temporal logic and probability, the frameworks presented in this thesis could be fruitfully applied, for instance, to domains such as automated medical applications (e.g. diagnosis and explanation), detection of complex activities (e.g. attacks on tcp/ip protocols, events from security cameras), and modelling of temporal phenomena (e.g. biological, geological phenomena). Some of these applications are surveyed in chapter 7.

The remainder of this section introduces example scenarios some of which are used as running examples throughout this thesis, with an emphasis on the medical domain which constitutes an appropriate application area for the epistemic capabilities of the frameworks presented here.

1.2.1 Medical Scenarios

As mentioned above, a suitable application domain for logic programming and RAA techniques is the *medical* one. For instance, parts of *Watson* (see e.g. [20]), developed at IBM, are written in *Prolog* – a logic programming language often used in the context of RAA. *Watson* has gained worldwide attention for winning in Natural Language-based TV show *Jeopardy!*, and it is now being applied to medicine and automated healthcare [21, 16]. The frameworks presented in this thesis, with their ability to deal with probabilistic data and knowledge, lend themselves to scenarios where patients undergo medical tests (modeled by imperfect sensing actions) and take action accordingly. The first scenario does not use any epistemic feature and is taken as a starting point.

Scenario 1.1 (Antibiotic)**.** A patient has a rash often associated with a bacterial infection, and can take an antibiotic known to be reasonably effective against the infection. Treatment is not always successful, and if not may still clear the rash. Failed treatment leaves the bacteria resistant. The patient is treated twice.

This scenario is a rather simplistic one, and does not correspond to what usually happens in real life in many ways – for example, the patient is treated twice with an antibiotic, regardless of any impact on her symptoms, putting herself under risk of the bacteria becoming resistant. In reality, medical tests are often employed to find out whether someone actually needs treatment. Nonetheless, this scenario involves a good amount of uncertainty as implied by the use

of expressions such as “often”, “reasonably”, “not always” and “may”, and therefore is a good benchmark problem for the non-epistemic framework introduced in this thesis.

A problem that involves medical testing is the following from [17]:

Scenario 1.2 (Breast Cancer Problem)**.** The probability of breast cancer is 1% for a woman at 40 who participates in a routine screening. If a woman has breast cancer, the probability is 79.2% that she will have a positive mammography. If a woman does not have breast cancer, the probability is 9.6% that she will also have a positive mammography. A woman in this age group had a positive mammography in a routine screening. What is the probability that she actually has breast cancer?

This example has been used to show that humans are not good at taking into account *base rates* (in this example, the base rate is the 1% probability that a woman has breast cancer when participating in a routine testing) when reasoning probabilistically. Indeed, it was found in [17] that about 95 out of 100 physicians (almost) discard any prior probability and estimate the actual probability to be around 75% – this dramatically differs from the actual probability of 7.7%. This systematic fallacy of human reasoning is known as *base rate neglect* and similar instances have been observed and extensively studied (see e.g. [35]). Given that humans (including experts) tend to misjudge probabilities in similar scenarios, frameworks introduced in this thesis could be useful, for example, to inform patients about the actual interpretation of their medical tests results and to aid medical decisions while providing explanations and correct estimates for probabilities based on the patients' medical histories.

The following is a more realistic scenario adapted from [1]:

Scenario 1.3 (Tuberculosis)**.** *Tuberculosis* (*TB* for short) is thought to affect 1/3 of people in the world, and every year a further 1% of the population is newly infected. It is mainly spread through its bacterial pathogen *Mycobacterium tuberculosis* via inhalation of infected droplets. Approximately 80–90% of individuals facing high level of exposure to the pathogen are infected, and infection is asymptomatic in most cases (*latent TB*) as only about 5% develop the clinincal disease (*active TB*) within a short amount of time following the infection. Individuals with latent TB do not transmit the disease, but they carry a 5% lifetime risk of developing active TB, mainly because of reactivation of the original pathogen. According to a particular patient's report, he was exposed to TB twice in the past, when he was 30 and when he was 32. The doctor follows some guidelines and judges the first exposure as a low-risk exposure and the second one as a high-risk one.

Scenario 1.3, which is summarised in fig. 1.1, can be readily modelled using frameworks presented in this thesis, and is used to show some of their advanced features such as *(probabilsitic) triggered actions*. Furthermore, it allows for extensions that would make it more realistic (for instance by taking into account that HIV-positive patients affected with latent TB carry a 10% yearly risk of reactivation, and that reactivation probability is significantly higher within 7 years from infection). Among these, the following extension makes use of epistemic reasoning:

Scenario 1.4 (Tuberculosis, Epistemic extension)**.** The doctor advises him to perform a *tuberculin skin test*, and then start treatment with antibiotics if he is highly likely infected with the pathogen to avoid further development of the disease.

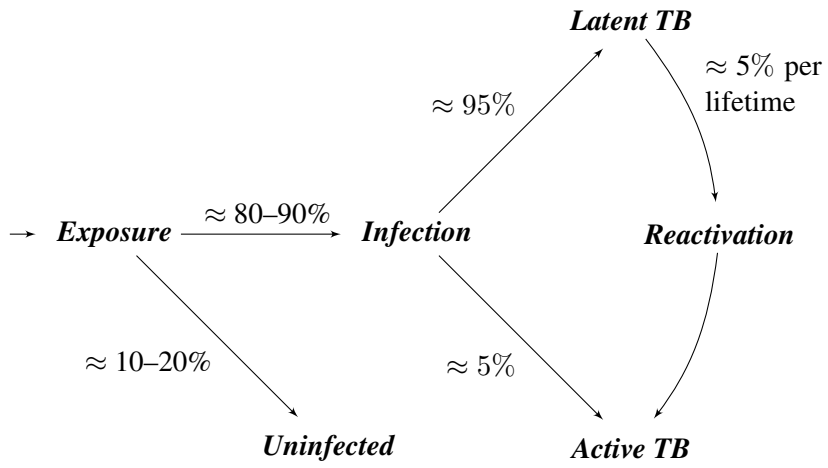


Figure 1.1: Diagrammatic representation of the effects of a single exposure in scenario 1.3, adapted from [1].

1.2.2 Other Scenarios

Section 1.2.1 presents scenarios that are closely related to the field of medicine and which immediately suggest an application area. However, it is useful to introduce other scenarios – not necessarily tied to a specific application domain – in order to show other features of the frameworks introduced in this thesis. These scenarios can be regarded as simple narratives and are discussed for different purposes and under different perspectives throughout the course of this thesis.

A simple coin-toss scenario is used to introduce some basic concepts:

Scenario 1.5 (Coin Toss)**.** A coin initially (instant 0) shows Heads. A robot can attempt to toss the coin, but there is a small chance that it will fail to pick it up, leaving the coin unchanged. The robot attempts to toss the coin (instant 1).

The following scenario can be seen as an example of *planning*:

Scenario 1.6 (Light switch)**.** A button is used to turn a light on/off. Due to a faulty contact, pressing the button does not always have an effect. An agent does not know if the light is initially on or off, and can only imperfectly sense if it is off or on. The goal is to have the light turned on at instant 2 onwards, hence the agent first imperfectly senses the light (instant 0) and then presses the button at a later time (instant 1) if reasonably certain that the light is off.

1.3 Thesis Contributions and Outline

This thesis' main contribution is to present a novel combination of several types of reasoning, some of which were discussed in section 1.1, namely :

1. Reasoning About Actions
2. Narrative Reasoning
3. Plausible Reasoning

4. Epistemic Reasoning

5. Support for Belief-Conditioned Actions

This resulted in PEC+ and EPEC, two *action languages* based on the *Event Calculus*' ontology which use *probability theory* as a representation for plausibility and belief. These frameworks fit in a current trend in logic-based AI and enrich the current set of available methods, e.g.: EFEC [42] (supporting 1, 2, 4 and 5), BHL [3] (supporting 1, 3 and 4), Language $\mathcal{E}$+ (supporting 1, 3, 4 and 5 but no imperfect sensing), and PAL [4], Prob-EC and MLN-EC [61, 62] (supporting 1, 2 and 3).

These languages are then implemented using two different paradigms (the *logic programming* and the *probabilistic programming* one). The correctness of the logic programming implementation is also proved.

This thesis is structured as follows:

- Chapter 2 introduces the context and some preliminary concepts that are needed for understanding the rest of the thesis.
- Chapter 3 introduces *PEC+* (short for *Probabilistic Event Calculus*), an action language combining a *narrative*-based approach to RAA with the ability to reason probabilistically.
- Chapter 4 implements *PEC+* in *ASP* (short for *Answer Set Programming*), a state-of-the-art logic programming language, and then proves and discusses some properties of this implementation, most notably its *correctness*. Some remarks about its scalability then follow.
- Chapter 5 introduces *EPEC* (short for *Epistemic Probabilistic Event Calculus*), an action language that combines narrative, probabilistic and epistemic reasoning. *EPEC* relies on PEC+ for its non-epistemic component.
- Chapter 6 uses approximate inference methods to tackle scalability problems of the implementation provided in chapter 4.
- Chapter 7 compare *PEC+* and *EPEC* to current related work.
- Chapter 8 draws some final conclusions with a view to further research.

Chapter 2

Background

This chapter introduces and discusses some background notions from first order logic, reasoning about actions, answer set programming and probability theory.

2.1 Reasoning About Actions

Reasoning About Actions (RAA for short) is a field at the intersection of Logic and AI whose main aim is that of modelling dynamical worlds where agent-controlled actions may happen. Its roots can be traced back to McCarthy [44], whose work led to the formalisation of one of the first RAA frameworks, the Situation Calculus [45]. However, the path leading to a sound formalisation of RAA was affected by problems [47, 31] whose solution required elaborating new techniques and sometimes even developing new logics [46, 58]. This chapter presents those techniques and problems which are relevant to this work.

2.1.1 Frame Problem

The *frame problem* can be briefly summarised as the problem of providing a concise representation of the principle that *most properties of the world do not change by virtue of an action being executed*. For instance, one might want to express the fact that, although opening a door might activate an alarm, the execution of such an action does not alter its colour, does not make one feel depressed, does not change the name of the current French prime minister and does not make an asteroid crash into Mars.

To see how this problem presents itself in practice, consider the following example:

Example 2.1 (Door Opening)**.** Consider a very simple classical logic language L consisting of the unary predicates *IsOpen*, *IsBlue* and *AttemptOpen*. The intended meaning of *IsOpen*(t) is that the door is open at time t, the intended meaning of *IsBlue*(t) is that the colour of the door is blue at time t and the intended meaning of *AttemptOpen*(t) is that an agent attempts to open the door at time t. In this setting, the two predicates *IsOpen* and *IsBlue* represent properties of the world and are usually called *fluents*, whereas *AttemptOpen* represents the occurrence of an *event* and is usually referred to as an *action*. At time zero, the door is shut and its colour is blue and the agent opens the door. It is reasonable to think that this situation is captured by the following axioms:

(D1) $\neg \mathit{IsOpen}(0)$,

(D2) $\mathit{IsBlue}(0)$,

(D3) $\forall t(\mathit{AttemptOpen}(t) \rightarrow \mathit{IsOpen}(t+1))$,

(D4) $\mathit{AttemptOpen}(0)$.

If two time constants $0, 1$ are considered (plus some axioms to deal with elementary arithmetic over $0, 1$) and every predicate symbol is interpreted as itself, theory $T = \{\mathit{D1}, \mathit{D2}, \mathit{D3}, \mathit{D4}\}$ has four models:

Model 1: $\neg \mathit{IsOpen}(0)$, $\mathit{IsBlue}(0)$, $\mathit{IsOpen}(1)$, $\mathit{AttemptOpen}(0)$, $\mathit{IsBlue}(1)$, $\mathit{AttemptOpen}(1)$,

Model 2: $\neg \mathit{IsOpen}(0)$, $\mathit{IsBlue}(0)$, $\mathit{IsOpen}(1)$, $\mathit{AttemptOpen}(0)$, $\mathit{IsBlue}(1)$, $\neg \mathit{AttemptOpen}(1)$,

Model 3: $\neg \mathit{IsOpen}(0)$, $\mathit{IsBlue}(0)$, $\mathit{IsOpen}(1)$, $\mathit{AttemptOpen}(0)$, $\neg \mathit{IsBlue}(1)$, $\mathit{AttemptOpen}(1)$,

Model 4: $\neg \mathit{IsOpen}(0)$, $\mathit{IsBlue}(0)$, $\mathit{IsOpen}(1)$, $\mathit{AttemptOpen}(0)$, $\neg \mathit{IsBlue}(1)$, $\neg \mathit{AttemptOpen}(1)$.

and in particular it is the case that $T \models \mathit{IsOpen}(1)$ but not that $T \models \mathit{IsBlue}(1)$ as desired. According to T, the door could have changed colour! Notice also that $T \not\models \mathit{AttemptOpen}(1)$ and $T \not\models \neg \mathit{AttemptOpen}(1)$, i.e. T is unable to determine whether another action *AttemptOpen* is executed at time 1.

To bridge the gap between intuition and such theories, McCarthy introduced the concept of *circumscription* [46], which formalises the idea that a predicate should be true for the fewest possible number of objects. This is done by transforming the formula θ to be minimised with respect to a predicate P to a second-order formula $\mathit{CIRC}[\theta, P]$. This new second order formula, informally speaking, says that there is no predicate Q which satisfies θ and is true of fewer objects than P, i.e. P is somewhat minimal with respect to truth: it only sets to true those objects which are required to be true by θ.

Example 2.2 (Door Opening, continued)**.** Recall theory $T = \{\mathit{D1}, \mathit{D2}, \mathit{D3}, \mathit{D4}\}$ and consider a new theory $T' = \mathit{CIRC}[T; \mathit{AttemptOpen}]$. It is now the case that $T' \models \neg \mathit{AttemptOpen}(1)$, i.e. the circumscription is forcing the default assumption that, although the theory carries no information about an occurrence of *AttemptOpen* at 1, *AttemptOpen* does not take place at time 1, which seems a reasonable assumption given the lack of information. Compare this with real world conversations: typically, when talking we purposely omit to mention the (infinite set of) actions that did not take place!

However, this new theory T' is still affected by the frame problem. It is indeed the case that $T' \not\models \mathit{IsBlue}(1)$ and $T' \not\models \neg \mathit{IsBlue}(1)$. Even worse, circumscribing with respect to the other predicates in the theory as well results in a nonsensical set of inferences, e.g.

$$\mathit{CIRC}[T; \mathit{AttemptOpen}, \mathit{IsBlue}, \mathit{IsOpen}] \models \mathit{IsBlue}(0)$$

and

$$\mathit{CIRC}[T; \mathit{AttemptOpen}, \mathit{IsBlue}, \mathit{IsOpen}] \models \neg \mathit{IsBlue}(1)$$

i.e., the door changes colour!

This happens because *change* has to be minimised rather than fluents, i.e. when possible, any fluent should keep its truth value as time flows. McCarthy's original intuition was to introduce new predicates to represent change in features of the world, and then minimise them. This can be done in the toy example by introducing two new predicates, *ChangeOpen* and *ChangeBlue*, and two new axioms

(D5) $\forall t[\mathit{ChangeOpen}(t) \leftrightarrow \neg(\mathit{IsOpen}(t) \leftrightarrow \mathit{IsOpen}(t+1))]$,

(D6) $\forall t[\mathit{ChangeBlue}(t) \leftrightarrow \neg(\mathit{IsBlue}(t) \leftrightarrow \mathit{IsBlue}(t+1))]$.

and applying circumscription to the new theory $T^+ = \{\mathit{D1}, \mathit{D2}, \mathit{D3}, \mathit{D4}, \mathit{D5}, \mathit{D6}\}$ yields

$$\mathit{CIRC}[T^+; \mathit{ChangeOpen}, \mathit{ChangeBlue}, \mathit{AttemptOpen}] \models \mathit{IsBlue}(1)$$

which finally meets intuition.

Although this might look like a definitive solution to the frame problem, it soon turns out that this is not necessarily the case as Hanks and McDermott point out in a classic paper [31], so that care has to be taken in how circumscription is applied to avoid anomalous models of theories. The next sections show how circumscription and analogous techniques (including Clark's completion [9] and Reiter's successor axioms [55]) are used to provide robust solutions to the frame problem by deriving default assumptions from theories.

Note that this thesis follows the usual convention in this field that all free variables in a theory are considered to be universally quantified with maximum scope.

Example 2.3 (Door opening, continued)**.** Using this convention, the example theory T^+ can be equivalently written as follows:

(D1) $\neg \mathit{IsOpen}(0)$,

(D2) $\mathit{IsBlue}(0)$,

(D3) $\mathit{AttemptOpen}(t) \rightarrow \mathit{IsOpen}(t+1)$,

(D4) $\mathit{AttemptOpen}(0)$,

(D5) $\mathit{ChangeOpen}(t) \leftrightarrow \neg(\mathit{IsOpen}(t) \leftrightarrow \mathit{IsOpen}(t+1))$,

(D6) $\mathit{ChangeBlue}(t) \leftrightarrow \neg(\mathit{IsBlue}(t) \leftrightarrow \mathit{IsBlue}(t+1))$.

2.1.2 The Situation Calculus

In the previous section actions and properties of the environment were naively axiomatised using simple unary first-order predicates. The *Situation Calculus* (SC for short) (originally introduced [45]; the version discussed here is adapted from [55]) is a more refined and systematic approach

to RAA, which is built on top of a *many-sorted*[1] second order calculus with standard axioms for equality. It is based on a branching time-structure, in which hypothetically occurring actions lead from one *situation* to another. Its main sorts are $\mathcal{S}$ for *situations*, $\mathcal{O}$ for objects, and $\mathcal{A}$ for actions. Informally speaking, a situation is a history of actions performed starting from the initial situation S_0. A binary function $Do : \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{S}$ is used to move from one situation to another, so that $Do(A, S)$ stands for the situation which is obtained by executing action A in situation S. A predicate $Poss(A, S)$ expresses the conditions under which A can be executed in situation S. Finally, a set of functions and relations called *functional and relational fluents* are used to model specific properties of the world. For example, fluents *WindowIsOpen* and *WaterIsHot* can be true or false according to whether the window is open or closed and similarly for the hot water. An action *AttemptCloseWindow* makes the fluent *WindowIsOpen* false, while an action *AttemptBoilWater* makes the fluent *HotWater* true.

Some everyday world features can be expressed using this language, as shown in the following example.

Example 2.4 (Door Opening in SC)**.** Consider the simple door example from Section 2.1.1. The fact that the door is initially open and blue can be expressed by the two axioms

(D1*) $IsOpen(S_0)$

(D2*) $IsBlue(S_0)$

while the action of opening the door might be modelled through

(D3*) $IsOpen(Do(AttemptOpen, s))$

where *IsOpen* and *IsBlue* are relational fluents, *AttemptOpen* is an action, and s is a situation variable (implicitly universally quantified).

This model can be extended to the case in which the door can be unlocked, the door is initially locked, unlocking the door only works when the door is closed, and an attempt to open the door does not succeed when the door is locked by simply adding the following axioms:

(D4*) $\neg IsLocked(Do(AttemptUnlock, s))$

(D5*) $IsLocked(S_0)$

(D6*) $Poss(AttemptOpen, s) \leftarrow \neg IsLocked(s)$

(D7*) $Poss(AttemptUnlock, s) \leftarrow \neg IsOpen(s)$

Although the theory $T^* = \{D1^*, D2^*, D3^*, D4^*, D5^*, D6^*, D7^*\}$ can be used to prove many implicit properties of the world such as $IsOpen(Do(AttemptOpen, Do(AttemptUnlock, S_0)))$, many others such as $IsBlue(Do(AttemptOpen, Do(AttemptUnlock, S_0)))$ cannot be derived, implying that this language is affected by the frame problem as well! To overcome it, SC must be structured into

[1] A *many-sorted calculus* is a logic in which objects of the language are divided into *sorts*, so that a variable of a given sort can only take values in the corresponding sort. It is similar to what happens in strongly typed programming languages.

a more principled framework. Reiter's intuition to solve the frame problem [54] is to transform the *effect axioms* in T^* (i.e. those expressing the effect of an action on the world: axioms *D3** and *D4** in this example) in a standardised form called *successor state axioms*. For instance, in the door example, the successor state axiom for *IsLocked* can be written in the successor state axiom form $\forall a \forall s[\textit{IsLocked}(\textit{Do}(a, s)) \leftrightarrow \textit{IsLocked}(s) \wedge a \neq \textit{AttemptUnlock}]$, which states that the door is locked if and only if was already locked in the previous situation and the last action was not an attempt to unlock it.

When Successor State Axioms are paired with a standard set of axioms for stating actions' preconditions (known as *Action Precondition Axioms*) and some domain independent axioms constitute what it is called a *Basic Action Theory* (*BAT* for short), a widely studied set of SC theories which exhibit some desirable properties such as a solution to the frame problem. Although they are not defined formally here, in the following example the Door Opening example is translated into a BAT.

Example 2.5 (Door Opening in SC, continued)**.** Axioms *D6** and *D7** can be transformed into Action Precondition Axioms by transforming the implication into an if and only if. Axioms *D3** and *D4** can be transformed to Successor State Axioms as follows: axiom *D3** becomes $\textit{IsOpen}(\textit{Do}(a, s)) \leftrightarrow a = \textit{AttemptOpen} \vee \textit{IsOpen}(s)$, and *D4** becomes $\neg\textit{IsLocked}(\textit{Do}(a, s)) \leftrightarrow a = \textit{AttemptUnlock} \vee (\neg\textit{IsLocked}(s) \wedge a \neq \textit{AttemptLock})$; finally, axioms *D1**, *D2** and *D5** are a suitable set of initial conditions. These axioms, when considered together with domain-independent axioms, constitute a Basic Action Theory.

2.1.3 The Event Calculus

The *Event Calculus* (EC for short), which was first introduced as a logic program in [38] and then reformulated in classical logic (see e.g. [49]), is another well established language for reasoning about actions and change. It consists of a set of axioms (typically including axioms for $\mathbb{N}$ or $\mathbb{R}$ as an explicit representation of the timeline), some of which are domain independent and serve to describe general principles relating to effects of actions, the frame and related problems, and the others which constitute the domain dependent part of the theory.

Similarly to the Situation Calculus, EC models a part of reality through the use of fluents, which can be initiated/terminated by performing particular actions. Unlike the Situation Calculus, which is based on a branching structure, EC is mainly *narrative-based*, meaning that it can describe a *narrative* of events, i.e. a possibly incompletely specified set of actually occurring events. This difference is exemplified in Figure 2.1.

Since this thesis is mainly aimed at extending the Event Calculus, this section introduces its main formal definition alongside the intuition behind it. This adaptation is based on a functional dialect of EC, introduced in [42].

Definition 2.1 (Domain Language for EC)**.** A *domain language for EC* is a sorted predicate calculus, with a sort $\mathcal{A}$ for *actions* (variables $a, a_1, a_2, \ldots, a', a'', \ldots$), a sort $\mathcal{F}$ for *fluents* (variables $f, f_1, f_2, \ldots, f', f'', \ldots$), a sort $\mathcal{T}$ for *time points* (variables $t, t_1, t_2, \ldots, t', t'', ...$) and a sort $\mathcal{V}$ for *values* of the domain (variables $x, x_1, x_2, \ldots, x', x'', \ldots$). It uses five core predicates/functions: *Happens* defined over $\mathcal{A} \times \mathcal{T}$, *ValueOf* of the form $\mathcal{F} \times \mathcal{T} \rightarrow \mathcal{V}$, *CausesValue*

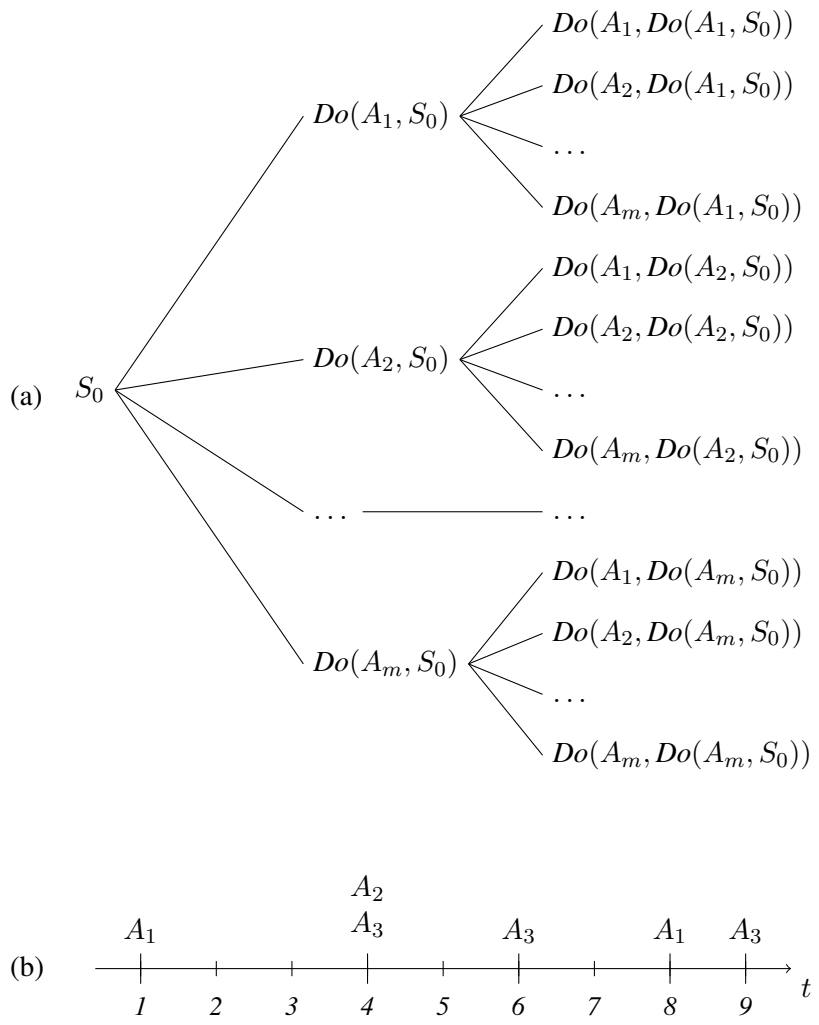


Figure 2.1: A comparison between SC and EC. (a) The Situation Calculus and its branching structure. In this example, exactly one out of m actions can occur in each situation, starting from S_0. As it has branching time, basic SC is suitable for representing hypothetical actions rather than actual ones. (b) The Event Calculus is narrative-based, and it can be visualised as a linear time line with actual (possibly concurrent) events occurring. In this example, three actions occur at distinct time points.

over $\mathcal{A} \times \mathcal{F} \times \mathcal{V} \times \mathcal{T}$, *PossValue* over $\mathcal{F} \times \mathcal{V}$ and $\leq$ defined over $\mathcal{T} \times \mathcal{T}$. In addition there are two auxiliary predicates: *ValueCaused* defined over $\mathcal{F} \times \mathcal{V} \times \mathcal{T}$ and *OtherValCausedBetween* defined over $\mathcal{F} \times \mathcal{V} \times \mathcal{T} \times \mathcal{T}$.

The intended meaning of *Happens*(A, T) is that action A is performed at time T. *ValueOf*$(F, T) = V$ means that the value of the fluent F at the time point T is V. *CausesValue* has a similar role to the one played by the pair *Initiates* and *Terminates* in prior versions of the EC. *CausesValue*(A, F, V, T) means that if action A is performed at time T then the fluent F is given cause to assume value V from that time point on. *PossValue* restricts the set of values that a fluent can take, i.e. *PossValue*(F, V) means that, in principle, F can take value V, $\leq$ is the usual ordering relation between instants. *ValueCaused*(F, V, T) means that some action happens at time T which gives cause for F to take value V. Finally, *OtherValCausedBetween*(F, V, T_1, T_2) means that some action occurs in the time interval $[T1, T2)$ which gives cause for F to take value other than V.

The five domain independent axioms are as follows:

Definition 2.2 (Domain Independent Axioms for EC)**.** The five *Domain Independent Axioms for EC* are:

(EC1) $\mathit{ValueCaused}(f, v, t) \leftrightarrow$
$\exists a(\mathit{Happens}(a, t) \wedge \mathit{CausesValue}(a, f, v, t))$,

(EC2) $\mathit{OtherValCausedBetween}(f, v, t_1, t_2) \leftrightarrow$
$\exists a \exists v'(\mathit{ValueCaused}(f, v', t) \wedge t_1 \leq t < t_2 \wedge v \neq v')$

(EC3) $\mathit{ValueOf}(f, t_2) = v \leftarrow$
$[(\mathit{ValueOf}(f, t_1) = v \vee \mathit{ValueCaused}(f, v, t_1)) \wedge$
$t_1 < t_2 \wedge \neg \mathit{OtherValCausedBetween}(f, v, t_1, t_2)]$

(EC4) $\mathit{ValueOf}(f, t_2) \neq v \leftarrow$
$[t_1 < t_2 \wedge \mathit{OtherValCausedBetween}(f, v, t_1, t_2) \wedge$
$\neg \exists t(t_1 \leq t < t_2 \wedge \mathit{ValueCaused}(f, v, t))]$

(EC5) $\mathit{PossVal}(f, t) \leftarrow \mathit{ValueOf}(f, t)$.

Axioms *EC1* and *EC2* are shorthand definitions. Axiom *EC3* expresses the idea that the a fluent has a particular value if it had the same value at an earlier time point and nothing happened in the meanwhile giving cause for its value to change. Axiom *EC4* captures the idea that a fluent does not have a particular value if it did not have that value at any earlier time point, and nothing happened in the meanwhile giving cause for it to change to that particular value. Finally, axiom *EC5* appropriately restricts the value that a fluent can take by linking the predicate *ValueOf* to information in the domain dependent part of the theory.

The following example shows how the domain dependent part of an EC theory can be realised:

Example 2.6 (Rolling a die)**.** In this example scenario, a die is rolled at time 2. To describe this, a single fluent symbol (*DieFaceShowing*) taking values in the set $\{1, 2, 3, 4, 5, 6\}$ and an action symbol (*AttemptRoll*) are sufficient. The domain dependent part of the theory consists of:

(R1) $v = 1 \vee v = 2 \vee v = 3 \vee v = 4 \vee v = 5 \vee v = 6$,

(R2) $\mathit{PossVal}(\mathit{DieFaceShowing}, v)$,

(R3) $\mathit{CausesValue}(\mathit{AttemptRoll}, \mathit{DieFaceShowing}, v, t)$,

(R4) $\mathit{Happens}(\mathit{AttemptRoll}, 2)$.

The theory $\mathit{CIRC}[\Delta; \mathit{CausesValue}] \wedge \mathit{CIRC}[\Omega; \mathit{Happens}] \wedge \mathit{CIRC}[\Lambda; \mathit{PossVal}] \wedge \Sigma$, where Δ is the conjunction of *CausesValue* formulas, Ω is the conjunction of *Happens* formulas, Λ is the conjunction of *PossVal* formulas and $\Sigma = \{\mathit{EC1}, \mathit{EC2}, \mathit{EC3}, \mathit{EC4}, \mathit{EC5}\}$ is the set of domain independent axioms for EC, has 6^2 models, one for each possible initial value of *DieFaceShowing* and one for each possible outcome of the *AttemptRoll* action.

2.1.4 Action Languages

An alternative to the use of classical logic in the context of reasoning about action has been the development of specialized action languages, with their own bespoke syntax and semantics. The first two such languages to be introduced were *STRIPS* [22] and *ADL* [52], followed by *language* $\mathcal{A}$ by Gelfond and Lifschitz [28].

The following sections briefly discuss Language $\mathcal{A}$, on which many of the modern probabilistic languages for reasoning about actions are based, as well as language $\mathcal{E}$ which is extended with probabilities in the present thesis.

Language $\mathcal{A}$

Language $\mathcal{A}$ was kept intentionally simple and, similarly to most action languages, it models a domain of the world using an appropriate set of propositions. Language $\mathcal{A}$ has a very simple language constituted by a set $\mathcal{F}$ of *fluents* and a set $\mathcal{A}$ of *actions*. Its syntax includes propositions of two kinds: *c-propositions* are used to state effects of actions and have the form

$$A \textbf{ causes } L \textbf{ if } L_1, L_2, \ldots, L_m$$

where $A \in \mathcal{A}$ is an action and $L, L_1, L_2, \ldots, L_m$ are *fluent literals* (i.e., they are either a fluent $F \in \mathcal{F}$ or its negation $\neg F$), with "A **causes** L" being shorthand for the case $m = 0$. *v-propositions* are used to state that a given literal is observed after a given sequence of actions, and have the form

$$L \textbf{ after } A_1, A_2, \ldots, A_m$$

where L is a fluent literal and $A_1, A_2, \ldots, A_m$ is a possibly empty sequence of actions, with "**initially** L" being shorthand for the case where $m = 0$.

Language $\mathcal{A}$'s simple syntax can be demonstrated through a classic example:

Example 2.7 (Yale Shooting Problem)**.** This example concerns a well-known scenario known as *the Yale Shooting Problem* [31] (*YSP* for short) using Language $\mathcal{A}$. YSP is mentioned above when talking about the frame problem (including Circumscription from section 2.1.1). It is also known as Hanks-McDermott Problem, after the two researchers that invented it to highlight flaws affecting some attempts to solve the frame problem. In YSP a gun is initially unloaded and a human target is initially alive. The gun is then loaded and it shoots at the target. A possible axiomatisation is as follows:

(YSP1) **initially** $\neg$*Loaded*,

(YSP2) **initially** *Alive*,

(YSP3) *Load* **causes** *Loaded*,

(YSP4) *Shoot* **causes** $\neg$*Alive* **if** *Loaded*,

(YSP5) *Shoot* **causes** $\neg$*Loaded*,

A set of such propositions is called a *domain description for Language $\mathcal{A}$*. The semantics of language $\mathcal{A}$ is given in terms of *states* and *transition functions*. Intuitively speaking, a *state* is a description of which fluents hold in the world and which do not, while the *transition function* specifies how states get updated when an action is performed. Starting from a domain description $\mathcal{D}$, Language $\mathcal{A}$'s semantics works out *models* of a domain description, each one being a pair $(S^0_{\mathcal{D}}, t_{\mathcal{D}})$ where $S^0_{\mathcal{D}}$ describes the initial state of the world (i.e., before any action is performed) according to $\mathcal{D}$ and $t_{\mathcal{D}}$ is a description of how states get updated according to the causal information in $\mathcal{D}$.

Example 2.8 (Yale Shooting Problem, continued)**.** Let $\mathcal{D}_Y = \{\mathit{YSP1}, \mathit{YSP2}, \mathit{YSP3}, \mathit{YSP4}, \mathit{YSP5}\}$. The only model $(S^0_{\mathcal{D}_Y}, t_{\mathcal{D}_Y})$ is the one specified by

$$S^0_{\mathcal{D}_Y} = \{\mathit{Alive}\},$$

$$t_{\mathcal{D}_Y}(\mathit{Load}, S) = S \cup \{\mathit{Loaded}\},$$

$$t_{\mathcal{D}_Y}(\mathit{Shoot}, S) = \begin{cases} S - \{\mathit{Loaded}, \mathit{Alive}\} & \text{if } \mathit{Loaded} \in S, \\ S & \text{otherwise.} \end{cases}$$

Since $\mathcal{D}_Y$ has exactly one model it is said to be *consistent* and *complete*.

Language $\mathcal{A}$ has a notion of entailment which is demonstrated in the following example:

Example 2.9 (Yale Shooting Problem, continued)**.** $\mathcal{D}_Y$ entails (among others) v-propositions "**initially** *Alive*, *Loaded* **after** *Load*" and "$\neg$ *Alive* **after** *Load*, *Shoot*".

Language $\mathcal{E}$

Language $\mathcal{E}$ [36] is an action language analogous to Language $\mathcal{A}$, but based on a different ontology. Indeed, while Language $\mathcal{A}$ is similar to the Situation Calculus, Language $\mathcal{E}$ is inspired by the Event Calculus, from which it inherits the capacity of dealing with narratives. Its language consists of a set $\mathcal{F}$ of fluents, a set $\mathcal{A}$ of actions, a set $\mathcal{T}$ of time points and a partial (possibly total) ordering $\leq$ between time points. Its syntax consists of three proposition types. *c-propositions* are used to specify actions' effects and have either the form

$$A \textbf{ initiates } F \textbf{ when } L_1, L_2, \ldots, L_m$$

or the form

$$A \textbf{ terminates } F \textbf{ when } L_1, L_2, \ldots, L_m$$

where $A \in \mathcal{A}$, $F \in \mathcal{F}$ and $L_1, L_2, \ldots, L_m$ are fluent literals, with "A **initiates** F" and "A **terminates** F" being shorthand for the case $m = 0$. Propositions to express event occurrences are called *h-propositions* in $\mathcal{E}$ and have the form

$$A \textbf{ happens-at } T$$

for $A \in \mathcal{A}$ and $T \in \mathcal{T}$. Finally, it is possible to express that a fluent literal holds at a given time-point through *t-propositions* of the form

$$L \textbf{ holds-at } T$$

where L is a fluent literal and $T \in \mathcal{T}$.

Any set of c-propositions, h-propositions and t-propositions is called a *domain description*.

Example 2.10 (Taking a Medicine)**.** A patient is initially ill and a medicine is known to cure this disease. A suitable domain description $\mathcal{D}_{TM}$ for this scenario is the following:

(TM1) *HasFlu* **holds-at** 0

(TM2) *TakeMedicine* **terminates** *HasFlu*

Similarly to Language $\mathcal{A}$, Language $\mathcal{E}$'s semantics defines *models* of a domain description. In this case, however, a model is a mapping $\mathcal{F} \times \mathcal{T} \rightarrow \{\top, \bot\}$ that describes the state of the world (in terms of which fluents are true) at every time point. A domain description that has exactly one model is said to be *consistent* and *complete*. Finally, a domain description $\mathcal{D}$ *entails* the v-proposition F **holds-at** T (also written $\mathcal{D} \models F$ **holds-at** T) iff for every model H of $\mathcal{D}$ $H(F, T) = \top$, and entails the v-proposition $\neg F$ **holds-at** T iff for every model H of $\mathcal{D}$ $H(F, T) = \bot$.

Example 2.11 (Taking a Medicine, continued)**.** $\mathcal{D}_{TM}$ entails *HasFlu* **holds-at** T for any time point T. Notice that if $\neg$*HasFlu* **holds-at** 1 is the goal, then {*TakeMedicine* **happens-at** 0} is a good *plan* with respect to the goal, because it is the case that $\mathcal{D}_{TM} \cup$ {*TakeMedicine* **happens-at** 0} $\models \neg$*HasFlu* **holds-at** 1.

Amongst the several extensions of Language $\mathcal{E}$ that have been developed along the years, Modular-$\mathcal{E}$ [37] is a modular, elaboration tolerant extension of Language $\mathcal{E}$ which is able to deal with the frame, qualification and ramification problems in a robust way. The syntax used to develop an Epistemic Probabilistic Event Calculus (EPEC) is inspired by that of Modular-$\mathcal{E}$.

2.1.5 Epistemic Reasoning

In the context of RAA, *Epistemic Reasoning* allows the possibility of dropping the somewhat unrealistic assumption that agents have perfect knowledge of the world by dealing with (possibly imperfect) *sensing* and (possibly incomplete or approximate) *states of knowledge*. For example, consider the generic modus ponens: From $A \rightarrow B$ and A one can infer B in plain propositional logic. However, if an agent (aware of the rule $A \rightarrow B$) has no direct access to the truth value of A, s/he cannot conclude B until A is somehow *sensed by* the agent. Such sensing actions are a core part of everyday life, especially from a planning perspective: to call my landlady, I need to know her telephone number first, and in order to find out I might ask someone who knows her; to know whether my car needs oil, I can check the oil gauge; and I can check the airport screens to know which gate my plane departs from.

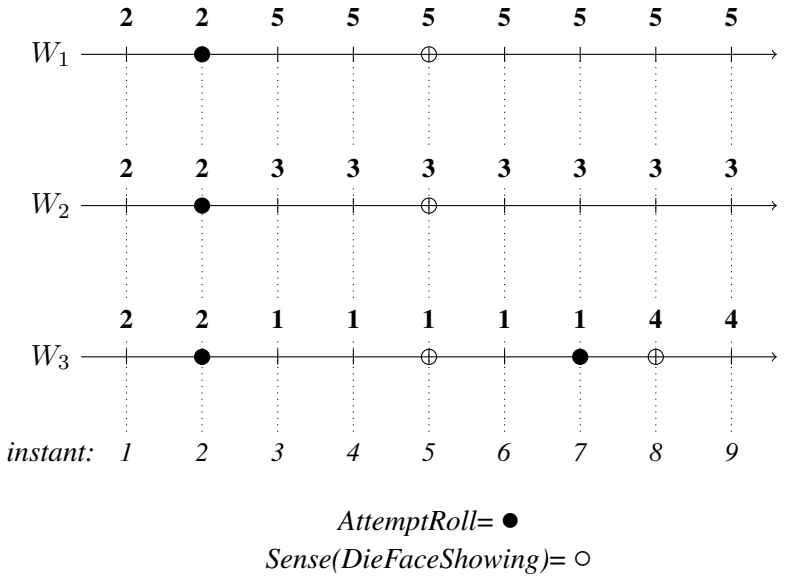
The need for knowledge producing (sensing) actions motivates the need for epistemic RAA languages. Knowledge-oriented reasoning about actions has its roots in the work of Moore [50],

whose work has been continued and extended in many directions. Amongst such contributions, Levesque and Scherl's situation calculus based work [56, 57] modelled states of knowledge through the use of possible situations and provided a solution to the frame problem in this context.

This section describes the *Epistemic Functional Event Calculus* [42] which introduces the ability to deal with partial states of knowledge in the (Functional) Event Calculus introduced in Section 2.1.3.

The Epistemic Functional Event Calculus

The *Epistemic Functional Event Calculus* [42] (*EFEC* for short) is based on the Functional Event Calculus and is a possible-worlds approach to epistemic reasoning, as it uses the notion of a *world* to represent the considered agent's current state of knowledge. Worlds can be thought of as timelines representing legal evolutions of the modelled environment. For instance, in a rolling die example (similar to Example 2.6) worlds can be visualised as follows:



where bold numbers represent the value of functional fluent *DieFaceShowing* at different time-points. An an action *AttemptRoll* is executed in all worlds and then the value of *DieFaceShowing* is sensed through a sensing action *Sense*(*DieFaceShowing*), but then only in W_3 action *AttemptRoll* is executed again. In world W_1, although at instant 3 the die is showing face 5 the agent has not yet sensed this and therefore believes that it could e.g. be in world W_1, W_2 or W_3.

EFEC inherits all the sorts and domain independent axioms from EC as in Section 2.1.3, plus two new sorts: a sort $\mathcal{W}$ for *worlds* and a sort $\mathcal{I}$ for *instants*[2] that are ordered with respect to a partial ordering relation $\preceq$. A mapping $\langle\rangle : \mathcal{W} \times \mathcal{I} \rightarrow \mathcal{T}$ is introduced such that $\langle W, I\rangle$ is meant to represent instant I in possible world W. A set of domain independent axioms ensures that every time point T can be expressed in the form $\langle W, I\rangle$ and gives conditions under which

[2]Notice that in EFEC instants from the sort $\mathcal{I}$ coexist with time-points from the sort $\mathcal{T}$. These two sorts should not be confused. The difference will be made clear later on.

it is possible to compare two time points under $<$, namely $\langle w, i\rangle = \langle w', i'\rangle$ when $w = w'$ and $i = i'$, and $\langle w, i\rangle \leq \langle w', i'\rangle$ when $w = w'$ and $i \preceq i'$.

The core of EFEC is a function K used to specified the agent's state of knowledge, hence called an *epistemic fluent*. Fluent $K(W)$ represents the *accessibility relation* of W, so that $\mathit{ValueOf}(K(W), \langle W', I\rangle) = \top$ means that W is accessible from W' at instant I. Accessibility relations are used in modal logics and can have different properties (e.g. reflexivity, symmetry, transitivity) according to what is being modelled by them (see e.g. [24]). In the case of EFEC, the accessibility relation models an agent's knowledge. In modal logic this is usually modelled through reflexive, symmetric and transitive equivalence relations, therefore some domain independent axioms in EFEC make sure that K is an equivalence relation. Special actions of the form $\mathit{Sense}(F)$ where $F \in \mathcal{F}$ represent the agent's sensing of the value of fluent $\mathcal{F}$. A domain independent axiom ensures that when sensing actions are performed by the modelled agent, they modify its accessibility relation by making inaccessible all those worlds having a different fluent value from that which the agent sensed.

EFEC includes predicates (and corresponding axioms) to represent and use knowledge gained from sensing actions. For instance, $\mathit{KnowsValueIsNot}(\langle W, I\rangle, F, I', V)$ expresses the fact that an agent in world W at time I knows that the value of F at instant I' will not be V: this is axiomatised by imposing that for every world W' accessible from W at instant I, the value of fluent F in W' at time I' is not V. Other predicates such as *KnowsValueIs*, *KnowsValue*, *KnowsHappens*, *KnowsNotHappens* and *KnowsIfHappens* are similar and their names self-explanatory. Actions in EFEC may be conditioned on *epistemic* preconditions, meaning that an action occurrence may depend on an agent's knowledge of a specific fluent. This is realised through a predicate *PerformIfValueKnownIs* and a corresponding axiomatisation. Predicates *Perform* and *Triggered* are also available for unconditional actions and triggered events respectively. Finally, EFEC's domain independent part ensures that the number of possible worlds is big enough as to correctly represent lack of knowledge about the world. For instance, if the initial value of N truth-valued fluents is unknown, at least 2^N worlds are needed, one for each possible truth assignment.

The following example demonstrates some of EFEC's features:

Example 2.12 (Taking Medicine, Epistemic Version)**.** A person who might have contracted a life threatening flu undergoes a test to find out. Since the available cure for this disease makes the patient's skin blue as a side effect, he decides to take the medicine only if he is actually diseased. A suitable domain dependent axiomatisation for this scenario is the following:

(TM1*) $\mathit{PossVal}(\mathit{HasFlu}, v) \leftrightarrow (v = \top \vee v = \bot)$,

(TM2*) $\mathit{PossVal}(\mathit{HasBlueSkin}, v) \leftrightarrow (v = \top \vee v = \bot)$,

(TM3*) $\mathit{KnowsValueIsNot}(\langle 0\rangle, f, 0, v) \leftrightarrow f = \mathit{HasBlueSkin} \wedge v = \top$,

(TM4*) $\mathit{CausesValue}(\mathit{TakeMedicine}, \mathit{HasFlu}, \bot, t)$,

(TM5*) $\mathit{CausesValue}(\mathit{TakeMedicine}, \mathit{HasBlueSkin}, \top, t)$,

(TM6*) $\mathit{Perform}(a, i) \leftrightarrow a = \mathit{Sense}(\mathit{HasFlu}) \wedge i = 1$,

(TM7*) $\mathit{PerformIfValueKnownIs}(a, f, v, i) \leftrightarrow a = \mathit{TakeMedicine} \wedge f = \mathit{HasFlu} \wedge v = \top \wedge i = 3$

where axioms *TM1** and *TM2** state that *HasFlu* and *HasBlueSkin* are truth-valued fluents, axiom *TM3** describes the fact that it is known that at time 0 the patient's skin is not blue, axioms *TM4** and *TM5** describe the effects of taking the medicine, and must be circumscribed together with EFEC's Sensing Axiom to address the frame problem. Axiom *TM6** states that the patient undergoes the test at time 1, and finally *TM7** describes the conditional action of taking the medicine at time 3 if the test is positive. Axioms for domain closure and Uniqueness Names Axioms are also needed.

Circumscribing this theory along with the domain independent axioms for EC and EFEC classically entails:

($\models$TM1*) $\mathit{KnowsValueIs}(\langle 0 \rangle, \mathit{HasFlu}, 4, \bot)$

($\models$TM2*) $\mathit{KnowsValue}(\langle 2 \rangle, \mathit{HasFlu}, 0)$

($\models$TM3*) $\neg \mathit{KnowsIfHappens}(\langle 0 \rangle, \mathit{TakeMedicine}, 3)$

($\models$TM4*) $\mathit{KnowsIfHappens}(\langle 2 \rangle, \mathit{TakeMedicine}, 3)$

($\models$TM5*) $\mathit{KnowsValue}(\langle 2 \rangle, \mathit{HasBlueSkin}, 4)$

($\models$TM6*) $\mathit{KnowsValueIs}(\langle 2 \rangle, \mathit{HasBlueSkin}, 2, \bot)$

Entailment $\models$*TM1** can be seen as the *goal* of not having the flu at time 4. Entailment $\models$*TM2** represents a form of reasoning about the past: at time 2 the agent will have sensed the value of *HasFlu*, hence he will also known whether he was diseased at time 0. Entailment $\models$*TM3** expresses uncertainty about the future: having not sensed *HasFlu* yet, the patient cannot know whether he will have to take the medicine or not at time 3; however, this lack of knowledge is curtailed as soon as *HasFlu* is sensed, as shown by entailment $\models$*TM4**. Finally, entailments $\models$*TM5** and $\models$*TM6** state that the patient knows at time 2 whether he is condemned to have a blue skin at time 4, although his skin is normal in the present.

2.1.6 Logic Programming

The languages introduced throughout this section are commonly implemented using a programming paradigm known as *logic programming*. Programming languages belonging to this paradigm, e.g. Prolog (see e.g. [8]) and Answer Set Programming (see e.g. [29]), have a number of desirable properties which make the implementation of RA frameworks very natural. Some of these features are discussed below.

In logic programming, a (propositional[3]) *program* is usually regarded as a set of *Horn clauses*, i.e. rules of the form

$$H \leftarrow B_1, B_2, \ldots, B_n$$

[3]For simplicity, this section is only concerned with the propositional case.

for literals $H, B_1, B_2, \ldots, B_n$, where H is called the *head* of the rule and $B_1, B_2, \ldots, B_n$ is called the *body* of the rule. When $n = 0$ the previous clause takes the form

$$H$$

which is called a *fact*.

As already discussed above, RAA frameworks naturally rely on some form of non-monotonic reasoning (for example, circumscription). In logic programming, non-monotonic behaviour is usually obtained through the use of *negation as failure* (*NF* for short) which, in a nutshell, consists in deriving *not* A from failure to derive A. For instance, from the following set of Horn clauses

$$\begin{array}{c} A \leftarrow B, \\ C \leftarrow D, \\ B. \end{array} \tag{2.1}$$

NF would allow the derivation of *not* C and *not* D as it is impossible to show that they are true on the basis of the program's clauses[4].

Although using only Horn clauses is convenient from a computational complexity viewpoint[5], many problems cannot be represented this way: hence the need of introducing (propositional) *normal logic programs*, which use extended rules of the form

$$H \leftarrow B_1, B_2, \ldots, B_n, \textit{not } C_1, \textit{not } C_2, \ldots, \textit{not } C_m.$$

For example, the following set of clauses is a normal logic program:

$$\begin{array}{c} Z \leftarrow \textit{not } X, \textit{not } Y, \\ W \leftarrow Z. \end{array} \tag{2.2}$$

As Clark has shown [9], it is possible to give a semantics to such programs which captures the meaning of NF in terms of the classical propositional negation operator $\neg$. This semantics, known as *completion semantics*, involves a syntactical transformation of the program: every *not* is transformed into $\neg$, every collection of $\leftarrow$s with the same head is transformed into a $\leftrightarrow$, and a formula $\neg A$ is added for each proposition A such that A is not in the head of a rule or is not a fact.

For instance, under completion semantics, (2.1) becomes:

$$\begin{array}{c} A \leftrightarrow B, \\ C \leftrightarrow D, \\ B, \\ \neg D. \end{array}$$

[4]Notice that the word *not* is used in place of the negation operator $\neg$ to differentiate between negation as failure and classical negation.

[5]The problem of satisfying a set of Horn clauses can be solved in linear time, whereas the more general problem of satisfying an unrestricted set of boolean formulas, sometimes known as *SAT*, is an NP-complete problem which, if the $P \neq NP$ conjecture is true, cannot be solved in polynomial time.

which classically entails $\neg D$ and $\neg C$ as expected, and (2.2) becomes:

$$\begin{gathered} Z \leftrightarrow \neg X, \neg Y, \\ W \leftrightarrow Z, \\ \neg X, \\ \neg Y. \end{gathered}$$

which classically entails Z and W, as expected since X and Y cannot be inferred from the program (2.2).

Prolog is a well known logic programming language whose inference mechanism, called *SLDNF resolution*, is consistent with completion semantics. It supports negation as failure, and therefore it is straightforward to turn a normal logic program into a Prolog program.

Example 2.13. To show how situation calculus theories can be implemented in Prolog, consider a simple scenario, very similar to Example 2.1 used to introduce the frame problem and the situation calculus itself, in which a door can be opened, closed, locked and unlocked. The Basic Action Theory is:

(D1**) $\mathit{IsClosed}(S_0)$,

(D2**) $\mathit{IsLocked}(S_0)$,

(D3**) $\mathit{Poss}(\mathit{AttemptUnlock}, s) \leftrightarrow \mathit{IsLocked}(s)$,

(D4**) $\mathit{Poss}(\mathit{AttemptLock}, s) \leftrightarrow \neg\mathit{IsLocked}(s) \wedge \mathit{IsClosed}(s)$,

(D5**) $\mathit{Poss}(\mathit{AttemptOpen}, s) \leftrightarrow \mathit{IsClosed}(s) \wedge \neg\mathit{IsLocked}(s)$,

(D6**) $\mathit{Poss}(\mathit{AttemptClose}, s) \leftrightarrow \neg\mathit{IsClosed}(s)$,

(D7**) $\mathit{IsClosed}(\mathit{Do}(a, s)) \leftrightarrow (a = \mathit{AttemptClose}) \vee (a \neq \mathit{AttemptOpen} \wedge \mathit{IsClosed}(s))$,

(D8**) $\mathit{IsLocked}(\mathit{Do}(a, s)) \leftrightarrow (a = \mathit{AttemptLock}) \vee (a \neq \mathit{AttemptUnlock} \wedge \mathit{IsLocked}(s))$.

where axioms *D1***, *D2*** describe the initial situation, axioms *D3***, *D4***, *D5*** and *D6*** are Action Precondition Axioms and axioms *D7***, *D8*** are Successor State Axioms for Relational Fluents.

To implement it, first consider the following set of Horn clauses:

$$\begin{gathered} \mathit{IsClosed}(S_0), \\ \mathit{IsLocked}(S_0), \\ \mathit{Poss}(\mathit{AttemptUnlock}, s) \leftarrow \mathit{IsLocked}(s), \mathit{IsClosed}(s), \\ \mathit{Poss}(\mathit{AttemptLock}, s) \leftarrow \mathit{not}\ \mathit{IsLocked}(s), \\ \mathit{Poss}(\mathit{AttemptOpen}, s) \leftarrow \mathit{IsClosed}(s), \mathit{not}\ \mathit{IsLocked}(s) \\ \mathit{Poss}(\mathit{AttemptClose}, s) \leftarrow \mathit{not}\ \mathit{IsClosed}(s), \\ \mathit{IsClosed}(\mathit{Do}(a, s)) \leftarrow a = \mathit{AttemptClose}, \\ \mathit{IsClosed}(\mathit{Do}(a, s)) \leftarrow \mathit{not}\ (a = \mathit{AttemptOpen}), \mathit{IsClosed}(s), \\ \mathit{IsLocked}(\mathit{Do}(a, s)) \leftarrow a = \mathit{AttemptLock}, \mathit{IsClosed}(s), \\ \mathit{IsLocked}(\mathit{Do}(a, s)) \leftarrow \mathit{not}\ (a = \mathit{AttemptUnlock}), \mathit{IsLocked}(s) \end{gathered}$$

which under completion semantics yields precisely $\{D1^{**}, D2^{**}, D3^{**}, D4^{**}, D5^{**}, D6^{**}, D7^{**}, D8^{**}\}$, which in turn can be straightforwardly transformed into the following Prolog program:

```
% Definition of the situation sort:
situation(s0).
situation( do(A,S) ) :- situation(S), action(A).

% Definition of the action sort:
action(attemptOpen).
action(attemptUnlock).
action(attemptLock).
action(attemptClose).

% Definition of executable situation:
executable(s0).
executable( do(A,S) ) :-
    situation(do(A,S)), poss(A,S), executable(S).

% Initial situation:
isClosed(s0).
isLocked(s0).

% Action Precondition Axioms:
poss( attemptOpen, S ) :-
    situation(S), isClosed(S), \+ isLocked(S).

poss( attemptUnlock, S ) :-
    situation(S), isLocked(S).

poss( attemptLock, S ) :-
    situation(S), \+ isLocked(S), isClosed(S).

poss( attemptClose, S) :-
    situation(S), \+ isClosed(S).

% Successor State Axioms:
isLocked( do(A,S) ) :-
    situation(do(A,S)), A=attemptLock, isClosed(S);

isLocked( do(A,S) ) :-
    situation(do(A,S)), \+ (A=attemptUnlock), isLocked(S).
```

```
isClosed( do(A,S) ) :-
    situation(do(A,S)), A=attemptClose.

isClosed( do(A,S) ) :-
    situation(do(A,S)), \+(A=attemptOpen), isClosed(S).
```

using Prolog's convention that variable names are represented by names starting with a capital letter, that negation as failure is represented by `\+`, and that implication is represented by `:-`.

The definition of executable situations is used to cut out those situations that cannot be reached by executing legal actions one after another, e.g. $Do(AttemptOpen, S_0)$ is not a legal situation as the door is locked in situation S_0.

Now that the Basic Action Theory is implemented, it is possible to query it. As an example, it is possible to ask whether the door is open after having unlocked it:

```
?- \+ isClosed(do(attemptUnlock,s0)).
```

which evaluates to `false`. Notice that asking whether the door is open after having performed an *AttemptOpen* action in S_0 results in the following:

```
?- \+ isClosed(do(attemptOpen,s0)).
true.
```

However, this situation is not executable:

```
?- executable(do(attemptOpen,s0)).
false.
```

This implementation can be used to find (executable) situations in which the door is open:

```
?- executable(S), \+ IsClosed(S).
```

Of course, there are infinitely many such situations, so the result of this query can go on indefinitely:

```
S=do(attemptOpen,do(attemptUnlock,s0));
S=do(attemptLock,do(attemptOpen,do(attemptUnlock,s0)));
S=do(attemptUnlock,
     do(attemptLock,do(attemptOpen,do(attemptUnlock,s0))));
...
```

A more complex query is the following:

```
 ?- executable(S), action(A),
      \+ isClosed(S), \+ isLocked(do(A,S)).
```

which looks for an executable situation S and an action A such that the door is opened in S, and it is unlocked in situation $do(A, S)$. This results again in an infinite set of answers:

```
S = do(attemptOpen, do(attemptUnlock, s0)),
A = attemptOpen;

S = do(attemptOpen, do(attemptUnlock, s0)),
A = attemptUnlock;

S = do(attemptOpen, do(attemptUnlock, s0)),
A = attemptClose;

S = do(attemptLock, do(attemptOpen, do(attemptUnlock, s0))),
A = attemptUnlock;
...
```

Prolog is not an entirely declarative language: for instance, the order in which clauses appear can influence heavily the mechanism evaluating the queries, and in addition some operators (such as the cut operator, `!`) which can modify its behaviour procedurally. For this reason, Prolog is sometimes said to be a *proof-theoretic language*, that is, a language based on a mechanism which tries to derive a *proof* of a given sentence, hence putting an emphasis on computation.

An alternative is represented by *model-theoretic languages*, which try to build a *model* of a theory in order to establish whether a given query logically follows from such theory. *Answer Set Programming* (*ASP* for short) is one such model-theoretic language, and is built upon a recently introduced semantics, known as *stable models semantics* [27], which is not discussed here in detail. ASP's syntax is very similar to Prolog's, but it also supports other useful constructs such as choice rules, cardinality constraints and aggregates [60, 19].

In the following, an ASP implementation of the Event Calculus (see Section 2.1.3) is provided: the translation from the domain independent axioms *EC1*, *EC2*, *EC3*, *EC4* and *EC5* to ASP is excerpted from [42].

```
time(0..maxtime).

lessThanEqualTo(T1, T2) :- time(T1), time(T2), T1 <= T2.

lessThan(T1, T2) :- lessThanEqualTo(T1, T2), T1 != T2.

% EC1
valueCaused(F, V, T) :-
    happens(A, T), causesValue(A, F, V, T).

-valueCaused(F, V, T) :-
    possVal(F, V), time(T), not valueCaused(F, V, T).

% EC2
otherValCausedBetween(F, V, T1, T2) :-
        possVal(F, V),  valueCaused(F, V_other, T),
```

```
        lessThanEqualTo(T1, T),
        lessThan(T, T2),
        V != V_other.

-otherValCausedBetween(F, V, T1, T2) :-
        possVal(F, V),
        lessThan(T1, T2),
        not otherValCausedBetween(F, V, T1, T2).

% EC3
:- valueOf(F, T2, V_other), valueOf(F, T1, V), lessThan(T1, T2),
   V_other != V, not otherValCausedBetween(F, V, T1, T2).

% EC4
valueCausedBetween(F, V, T1, T2) :-
        valueCaused(F, V, T),
        lessThanEqualTo(T1, T),
        lessThan(T, T2).

-valueCausedBetween(F, V, T1, T2) :-
        lessThan(T1, T2),
        possVal(F, V),
        not valueCausedBetween(F, V, T1, T2).

:- valueOf(F, T2, V), lessThan(T1, T2),
   otherValCausedBetween(F, V, T1, T2),
   not valueCausedBetween(F, V, T1, T2).

% EC5
:- valueOf(F, T, V), not possVal(F, V).

% AUX1
% make sure valueOf is a function.
1 { valueOf(F, T, V) : possVal(F, V) } 1 :-
        fluent(F),time(T).
```

Amongst the peculiarities of ASP's syntax, notice the use of "–" for classical negation and the use of `not` for negation as failure; also, *AUX1* ensures that $valueOf$ is a function through the use of a particular ASP construct known as *choice rule*: `x{ atom_1, atom_2, ..., atom_n }y :- body` means that when `body` is satisfied, then some subset of `{ atom_1, atom_2, ..., atom_n }` having cardinality between `x` and `y` must be included in the stable model as well. The syntax `l:l1 : l2 : ... : ln`, for literals `l1`, `l2`, ..., `ln` is used to refer to the elements in

the set $\{\texttt{l} \mid \texttt{l1}, \texttt{l2}, \ldots, \texttt{ln}\}$, so for instance

```
r(a). r(b). r(c).
1{ p(a,X) : r(X) }1.
```

reduces to `1{p(a,a), p(a,b), p(a,c)}1`.

As an example of a problem which can be solved by EC once implemented, consider again Example 2.6 of a rolling die:

Example 2.14. Consider the theory $\{R1, R2, R3, R4\}$ from example 2.6 and its circumscription (see Section 2.1.1 for more details on circumscription). This is equivalent to the following ASP implementation:

```
#const maxtime=3.

fluent(dieFaceShowing).
action(attemptRoll).

% R1, R2
possVal(dieFaceShowing, 1..6).

% R3
causesValue(attemptRoll, dieFaceShowing, V, T) :-
        time(T),
        possVal(dieFaceShowing, V).

% R4
happens(attemptRoll, 2).
```

When queried together with the domain independent part of FEC using Clingo [26], a popular answer set solver and grounder, this program will terminate by saying that 36 models of the given theory were generated:

```
$ clingo 0 fec.lp rollingDie.lp
clingo version 4.5.3
Reading from fec.lp...
Solving...

SATISFIABLE

Models       : 36
Calls        : 1
Time         : 0.006s (Solving: 0.00s 1st Model: 0.00s
                       Unsat: 0.00s)
CPU Time     : 0.000s
```

2.2 Probabilistic Reasoning

2.2.1 The basics

Probability theory can be regarded as a scientific theory of dealing with events whose outcomes cannot be predicted with certainty. A central concept in probability is that of a *random variable*. A random variable takes value in a given (possibly continuous) *domain* with a given *probability*. A random variable's domain can be intuitively thought of as a set of possible outcomes of an experiment. As this thesis is mainly concerned with discrete random variables, in the following it is assumed that the domain of random variables is a discrete or finite set. Conventionally, random variables are denoted by capital letters $X, Y, \ldots$, their domains are denoted by curly upper case letters $\mathcal{X}, \mathcal{Y}, \ldots$, and their values by lower case letters $v, w, \ldots$ and the probability of a random variable X taking value v in its domain is typically written $p(X = v)$. It is common practice to write $p(X)$ to denote the probability distribution over values in the domain of X, and when the reference to the random variable X is clear from the context, it is possible to write $p(v)$ for $p(X = v)$. For instance, the fact that a random variable C associated with the outcome of a coin flip has probability $1/2$ to take value h (for heads) can be written $p(C = h) = 1/2$ or equivalently $p(h) = 1/2$. The function p defines a *probability distribution* over a random variable, and it is subject to the constraint

$$\sum_{v \in \mathcal{X}} p(v) = 1 \tag{2.3}$$

where $\mathcal{X}$ is the domain of X. It is said that p is a *probability distribution over* $\mathcal{X}$.

When many random variables are considered, $X_1, \ldots, X_n$ say, their *joint probability distribution* is written as $p(X_1, \ldots, X_n)$, and is subject to the constraint

$$\sum_{v_1 \in \mathcal{X}_1, \ldots, v_n \in \mathcal{X}_n} p(v_1, \ldots, v_n) = 1 \tag{2.4}$$

where $\mathcal{X}_1, \ldots, \mathcal{X}_n$ are the domains of $X_1, \ldots, X_n$ respectively. It is said that p is a *probability distribution over* $\mathcal{X}_1, \ldots, \mathcal{X}_n$.

For example, if X_1, X_2, X_3 represent the outcome of 3 distinct coin flips, then $p(h, t, t)$ denotes the probability of getting $X_1 = h$ and $X_2 = t$ and $X_3 = t$. Notice that the probability of $X_1 = h$ alone, in this example, can be intuitively calculated from the joint probability distribution as $\sum_{x,y} p(h, x, y) = p(h, h, h) + p(h, h, t) + p(h, t, h) + p(h, t, t)$, which expresses the intuitive fact that the probability of X_1 to take value h should be equal to the sum of the probabilities of getting $X_1 = h$ with X_2 and X_3 taking any of their possible values. This property, in a setting with $n + 1$ random variables $X, Y_1, \ldots, Y_n$, can be generalised to:

$$p(X = x) = \sum_{y_1, \ldots, y_n} p(X = x, Y_1 = y_1, \ldots, Y_n = y_n) \tag{2.5}$$

where the sum ranges over all possible values of $y_1, \ldots, y_n$. This is called the *sum rule* or *marginalisation rule*, and $p(X = x)$ is called the *marginal probability*.

A definition which plays a central role in this setting is that of *conditional probability*. $X =$

x *conditioned on* $Y = y$ or $X = x$ *given* $Y = y$ is written $p(X = x \mid Y = y)$. To understand what this stands for, consider an urn containing two white balls and three black balls, and an experiment which consists in blindly picking balls from that urn, without replacement. Two random variables X_1, X_2 represent the probability of picking a white or black ball at the first and second extraction respectively. The probability of picking a black ball on the first extraction is intuitively given by *favourable cases/total cases*, which is $3/5$ in this case. However, at the second extraction, the number of favourable cases and total cases has changed according to the result of first extraction. Hence, the probability of picking a black ball on the second extraction having picked a white ball on the first extraction is $3/4$, as the urn would be composed by 3 black balls and 1 white ball after the first extraction, while for similar reasons it would be $1/2$ if a black ball was extracted instead. This can be written $p(X_2 = b \mid X_1 = w) = 3/4$ and $p(X_2 = b \mid X_1 = b) = 1/2$. Notice that the probability of extracting first a black ball and then a white ball can be expressed as the probability of picking a black ball first and then a white ball *given* that a black ball was picked at first attempt. This can be written $p(X_1 = b, X_2 = w) = p(X_2 = w \mid X_1 = b)\, p(X_1 = b)$, which in the case with n random variables $X_1, \ldots, X_n$ can be generalised to

$$p(X_1 = x_1, \ldots, X_n = x_n) = \prod_{k=1}^{n} p(X_k = x_k \mid X_1 = x_1, \ldots, X_{k-1} = x_{k-1}) \tag{2.6}$$

which is called the *product rule*.

When $n = 2$, (2.6) can be rearranged to get

$$p(X = x \mid Y = y) = \frac{p(X = x, Y = y)}{p(Y = y)} \tag{2.7}$$

if $p(Y = y) \neq 0$, and using the product rule again together with the symmetry property $p(X, Y) = p(Y, X)$ to represent the numerator as $p(Y = y \mid X = x)p(X = x)$ and the sum rule to represent the denominator as $\sum_{x'} p(Y = y \mid X = x')p(X = x')$ yields:

$$p(X = x \mid Y = y) = \frac{p(Y = y \mid X = x)p(X = x)}{\sum_{x'} p(Y = y \mid X = x')p(X = x')} \tag{2.8}$$

which is the fundamental *Bayes' theorem*. Notice that when the value taken by a given random variable X cannot affect the value taken by another random variable Y, as it happens in the case of two distinct coin flips, conditioning X on Y does not change the distribution over X, i.e.

$$p(X \mid Y) = p(X) \tag{2.9}$$

which, using Bayes' theorem, implies $p(X, Y) = p(X)\, p(Y)$. If two random variables X and Y satisfy this condition, then they are said to be *independent*.

2.2.2 Probability and Logic

Although it might seem strange to merge logic (which is qualitative, deductive and deals with certain truths) with Probability Theory (which is quantitative and deals with uncertainty), it is

possible to show that the latter can be used to provide an extended semantics for the former, also preserving its classical semantics as a particular case. One well studied attempt can be found in [51]; it considers a propositional language L and defines a *probability function over sentences of* L as follows:

Definition 2.3 (Probability Function over L)**.** Consider a language L and the relative set SL of sentences of L (i.e., the closure of L under the propositional operators). A *probability function over a propositional language* L is a function $p : SL \to [0, 1]$ such that the following properties hold for any θ, ψ in SL:

(P1) If $\models \theta$ then $p(\theta) = 1$,

(P2) If $\theta \models \neg\psi$ then $p(\theta \vee \psi) = p(\theta) + p(\psi)$.

where $\models$ is the classical propositional logical consequence relation. For $p(\psi) \neq 0$, we also define the corresponding *conditional probability function* as

$$p(\theta \mid \psi) = \frac{p(\theta \wedge \psi)}{p(\psi)}.$$

From this definition, one can infer the following properties:

Proposition 2.1 (Properties of Probability Function)**.** For any θ, ψ in SL the following hold:

1. $p(\neg\theta) = 1 - p(\theta)$,
2. If $\models \theta$ then $p(\neg\theta) = 0$,
3. If $\theta \models \psi$ then $p(\theta) \leq p(\psi)$,
4. If $\models \theta \leftrightarrow \psi$ then $p(\theta) = p(\psi)$,
5. $p(\theta \vee \psi) = p(\theta) + p(\psi) - p(\theta \wedge \psi)$.
6. If $p(\psi) \neq 0$, then $p(\cdot \mid \psi) : SL \to [0, 1]$ is a probability function.

Proof. See [51, Proposition 2.2]. □

Notation 2.1. In the remainder of this thesis, the expression *probability function* is used for a function satisfying axioms *P1* and *P2* from definition 2.3, whereas *(joint) probability distribution* is used to indicate a function as described in section 2.2.1 satisfying constraints (2.3) and (2.4).

2.2.3 Probability as Belief

In Section 2.2.1 we made an appeal to intuition when defining the basic rules of probability theory. The motivating examples there are mainly based on a naive understanding of probability as the proportion of number of favourable cases over the number of total cases. However, this interpretation is somewhat narrow, and requires a number of intuitive assumptions (e.g., the cases must be intuitively mutually exclusive and equiprobable). A broader conception defines

probability as a specification of the frequency at which an *event*[6] occurs in a high number of trials, i.e. if we let n be the number of repetitions of an experiment, and E_n the number of times an event occurred in those n trials, then the frequentist interpretation of probability states that $p(E)$ converges to the ratio E_n/n as n gets bigger. This is somewhat exemplified by the expression

$$p(E) = \lim_{n \to \infty} \frac{E_n}{n}$$

where, however, one should be careful not to interpret the limit in its formal, mathematical sense, but just as a symbol to express the intuitive, informal concept expressed above.

This interpretation, however, is only meaningful when it is possible to repeat an experiment in very similar conditions. For this reason, the *Bayesian interpretation* of probability, which sees an event's probability as a measure of a (rational) agent's degree of belief on the event itself, can be seen as a further step in this context.

Several justifications for identifying belief and probability have been proposed and we discuss here a few popular ones.

The first justification is due to Cox [10], and it shows that if a belief function $Bel : SL \rightarrow [0, 1]$ over a set of propositional sentences of a language SL satisfies the following very reasonable desiderata:

A: Degrees of belief are represented by real numbers,

B: Degrees of belief correspond to common sense,

C: Degrees of belief are handled consistently.

then such belief function is forced in a sense to be a probability function p satisfying axioms *P1* and *P2* of definition 2.3, and, on the reception of new knowledge, the correspondent conditional belief function is forced to be the conditional probability function associated with p.

A second justification is provided by *Dutch Book Arguments* [53, 12], which show that an agent adopting a probability function p (which satisfies axioms *P1* and *P2*) of definition 2.3 as its measure of belief on propositional sentences *cannot be Dutch Booked*, i.e. it can never accept an *unfair* bet of the kind "If θ is true you lose 10 pounds, while if θ is false you owe me 10 pounds", which would surely result in a win for the bettor[7].

[6]The reader should be aware of the difference between an *event* in a probabilstic setting and that of an *event* in Reasoning about Actions. While in Probability Theory an event is a set of possible outcomes of an experiment, in Reasoning About Actions formalisms as the Event Calculus, events are elements of a narrative, which occur along the time line.

[7]National lotteries and football pools are tuned in a way such that the resulting bets are unfair to the bettors. For this reason, Bruno de Finetti was used to refer to Lotto, a popular Italian national lottery, as a "tax upon stupidity", as all the Lotto players are explicitly Dutch Booked by the bookmakers – see e.g. [13, Chapter 6, Page 65].

Chapter 3

Language PEC+

In this chapter the syntax and semantics of PEC+, the non-epistemic variant of the probabilistic RAA framework developed during this research, is defined.

With respect to the thesis contributions outlined in section 1.3, PEC+ combines Reasoning About Actions, Narrative Reasoning and Uncertain (Probabilistic) Reasoning, and therefore it is similar to previous work such as PAL [4], Prob-EC and MLN-EC [61, 62]. However, in addition to these languages, PEC+ adds support to triggered actions and will be shown to have a sound and complete implementation in chapter 4. It is also worth noting that none of these other languages offers *simultaneous* support for probabilistic events and probabilistic causal rules.

PEC+ is an extension of PEC, which was previously introduced in [11]. In the the context of this thesis it is mainly used as a "building block" for EPEC, which is introduced in chapter 5.

3.1 Syntax

Definition 3.1 (Domain Language)**.** A *domain language* for PEC+ is a tuple $\mathcal{L} = \langle \mathcal{F}, \mathcal{A}, \mathcal{V}, \mathit{vals}, \mathcal{I}, \leq, \bar{0} \rangle$ consisting of a finite non-empty set $\mathcal{F}$ of *fluents*, a finite set $\mathcal{A}$ of *actions*, a finite non-empty set $\mathcal{V}$ of *values* such that $\{\top, \bot\} \subseteq \mathcal{V}$, a function $\mathit{vals} : \mathcal{F} \cup \mathcal{A} \to 2^{\mathcal{V}} \setminus \emptyset$, a non-empty set $\mathcal{I}$ of *instants* and a minimum element $\bar{0} \in \mathcal{I}$ w.r.t. a total ordering $\leq$ over $\mathcal{I}$. For $A \in \mathcal{A}$ it is imposed that $\mathit{vals}(A) = \{\top, \bot\}$.

Example 3.1. Scenario 1.5 can be modelled using a language

$$\mathcal{L}_C = \langle \mathcal{F}_C, \mathcal{A}_C, \mathcal{V}_C, \mathit{vals}_C, \mathbb{N}, \leq_{\mathbb{N}}, 0 \rangle$$

where

$$\begin{gathered} \mathcal{F}_C = \{\mathit{Coin}\}, \\ \mathcal{A}_C = \{\mathit{Toss}\}, \\ \mathcal{V}_C = \{\top, \bot, \mathit{Heads}, \mathit{Tails}\}, \\ \mathit{vals}_C(\mathit{Coin}) = \{\mathit{Heads}, \mathit{Tails}\} \end{gathered}$$

$\mathbb{N}$ is the set of natural numbers (including 0), and $\leq_{\mathbb{N}}$ is the standard total ordering between naturals.

Example 3.2. Scenario 1.1 could be captured by a language

$$\mathcal{L}_A = \langle \mathcal{F}_A, \mathcal{A}_A, \mathcal{V}_A, \mathit{vals}_A, \mathbb{N}, \leq_{\mathbb{N}}, 0 \rangle$$

where

$$\begin{gathered}
\mathcal{F}_A = \{\mathit{Bacteria}, \mathit{Rash}\}, \\
\mathcal{A}_A = \{\mathit{TakesMedicine}\}, \\
\mathcal{V}_A = \{\top, \bot, \mathit{Weak}, \mathit{Present}, \mathit{Resistant}, \mathit{Absent}\}, \\
\mathit{vals}_A(\mathit{Bacteria}) = \{\mathit{Weak}, \mathit{Resistant}, \mathit{Absent}\}, \\
\mathit{vals}_A(\mathit{Rash}) = \{\mathit{Present}, \mathit{Absent}\}
\end{gathered}$$

Example 3.3. An appropriate domain language for scenario 1.3 is

$$\mathcal{L}_T = \langle \mathcal{F}_T, \mathcal{A}_T, \mathcal{V}_T, \mathit{vals}_T, \mathbb{N}_{\leq 50}, \leq_{\mathbb{N}_{\leq 50}}, 0 \rangle$$

where

$$\begin{gathered}
\mathcal{F}_T = \{\mathit{Tuberculosis}\}, \\
\mathcal{A}_T = \{\mathit{Exposure}\}, \\
\mathcal{V}_T = \{\top, \bot, \mathit{Absent}, \mathit{Latent}, \mathit{Active}\}, \\
\mathit{vals}_T(\mathit{Tuberculosis}) = \{\mathit{Absent}, \mathit{Latent}, \mathit{Active}\}, \\
\mathbb{N}_{\leq 50} = \{0, 1, \ldots, 50\}
\end{gathered}$$

and $\leq_{\mathbb{N}_{\leq 50}}$ is the total order between naturals equal or smaller than 50. The set of instants is taken to represent the number of years passed from the first exposure reported by the patient.

In what follows, all definitions are with respect to a domain language $\mathcal{L} = \langle \mathcal{F}, \mathcal{A}, \mathcal{V}, \mathit{vals}, \mathcal{I}, \leq, \bar{0} \rangle$.

PEC+ includes (fluent) literals and (fluent) formulas that are defined below. Literals and formulas with a time information attached are called i-literals and i-formulas respectively.

Definition 3.2 (Fluent and Action Literals, i-Literals)**.** A *fluent literal* is an expression of the form $F = V$ for some $F \in \mathcal{F}$ and $V \in \mathit{vals}(F)$. A fluent is *boolean* if $\mathit{vals}(F) = \{\top, \bot\}$. An *action literal* is either $A = \top$ or $A = \bot$. When no ambiguity can arise, $Z = \top$ is sometimes abbreviated to Z and $Z = \bot$ is abbreviated to $\neg Z$ for Z a fluent or action. An *i-literal* is an expression of the form $[L]@I$ for some (fluent or action) literal L and some $I \in \mathcal{I}$.

Definition 3.3 (Formulas, Fluent Formulas, i-Formulas)**.** The set of *formulas*, denoted by Θ, is the closure of the set of literals under $\wedge$ and $\neg$ (with $\vee$ and $\rightarrow$ being defined as shorthand in the usual way). A formula θ is said to be a *fluent formula* if it is formed from the set of fluent literals by closure under $\wedge$ and $\neg$. The set of *i-formulas*, denoted by Φ, is the closure of the set of i-literals under $\wedge$ and $\neg$. The shorthand $[\theta]@I$ stands for the i-formula formed from the formula θ and the instant I by replacing all literals L occurring in θ by $[L]@I$, e.g. $[F = V \rightarrow F' = V']@3$ is a shorthand for $[F = V]@3 \rightarrow [F' = V']@3$. When no ambiguity can arise, $\top$ stands for some fixed tautological formula.

Example 3.4. In scenario 1.5 the i-literal $[\mathit{Coin} = \mathit{Heads}]@3$ indicates that the coin shows heads at instant 3, while $[\neg \mathit{Toss}]@2$ indicates that the robot does not attempt to toss the coin at instant

2. In scenario 1.1, $[\mathit{Rash}{=}\mathit{Present}]@0 \wedge [\mathit{Bacteria}{=}\mathit{Absent} \wedge \mathit{TakesMedicine}]@3$ indicates that the patient initially has a rash, and that she takes the medicine and the bacterial infection is absent at instant 3.

Definition 3.4 (State, Partial State, Fluent State)**.** A *state* S is a set of literals, exactly one for each $F \in \mathcal{F}$ and $A \in \mathcal{A}$. A *partial state* is a subset $X \subseteq S$ of a state S. Given a partial state X, its subset containing all and only the fluent literals in X is a *partial fluent state*, and is denoted by $X{\restriction}\mathcal{F}$. For S a state, $S{\restriction}\mathcal{F}$ is called a *fluent state*. The subset of X containing all and only the action literals in X is denoted by $X{\restriction}\mathcal{A}$. The set of all states is denoted by $\mathcal{S}$, the set of all partial states is denoted by $\mathcal{X}$. Finally, the sets $\{S{\restriction}\mathcal{F} \mid S \in \mathcal{S}\}$ and $\{X{\restriction}\mathcal{F} \mid X \in \mathcal{X}\}$ are denoted by $\tilde{\mathcal{S}}$ and $\tilde{\mathcal{X}}$ respectively.

Example 3.5. One of the states that can be built with the elements of the domain language $\langle \mathcal{F}_A, \mathcal{A}_A, \mathcal{V}_A, \mathit{vals}_A, \mathbb{N}, \leq_{\mathbb{N}}, 0\rangle$ for scenario 1.1 is $S^1_A = \{\mathit{Bacteria}{=}\mathit{Resistant}, \mathit{Rash}{=}\mathit{Absent}, \neg\mathit{TakesMedicine}\}$. Its associated fluent state is $S^1_A{\restriction}\mathcal{F} = \{\mathit{Bacteria}{=}\mathit{Resistant}, \mathit{Rash}{=}\mathit{Absent}\}$. Any arbitrary subset of S^1_A, e.g. $X^1_A = \{\mathit{Rash}{=}\mathit{Absent}, \neg\mathit{TakesMedicine}\}$, is a partial state, whereas any arbitrary subset of $S^1_A{\restriction}\mathcal{F}$, e.g. $X^1_A{\restriction}\mathcal{F} = \{\mathit{Rash}{=}\mathit{Absent}\}$, is a partial fluent state.

Definition 3.5 (Outcome, Projection Functions)**.** An *outcome* is a pair of the form $(\tilde{X}, P^+)$ where $P^+ \in (0, 1]$. The two *projection functions* χ and π are such that for any outcome $O = (\tilde{X}, P^+)$, $\chi(O) = \tilde{X}$ and $\pi(O) = P^+$. The set of all outcomes $\tilde{\mathcal{X}} \times (0, 1]$ is denoted by $\mathcal{O}$.

Definition 3.6 (Weight of a Set of Outcomes)**.** Given a finite set of outcomes

$$B = \{O_1, O_2, \ldots, O_m\}$$

the *weight of* B is defined as

$$\pi(B) = \sum_{i=1}^{m} \pi(O_i).$$

Notation 3.1. The remainder of this thesis generally uses the following notation:

$I, I', I_1, I'', I_2, \ldots$ to denote elements of $\mathcal{I}$,
$A, A', A_1, A'', A_2, \ldots$ to denote elements of $\mathcal{A}$,
$F, F', F_1, F'', F_2, \ldots$ to denote elements of $\mathcal{F}$,
$V, V', V_1, V'', V_2, \ldots$ to denote elements of $\mathcal{V}$,
$\theta, \theta', \theta_1, \theta'', \theta_2, \ldots$ to denote formulas,
$\varphi, \varphi', \varphi_1, \varphi'', \varphi_2, \ldots$ to denote i-formulas,
$P, P', P_1, P'', P_2, \ldots$ to denote real values in $[0, 1]$,
$P^+, P^+_1, P^+_2, \ldots$ to denote real values in $(0, 1]$,
$S, S', S_1, S'', S_2, \ldots$ to denote elements of $\mathcal{S}$,
$X, X', X_1, X'', X_2, \ldots$ to denote elements of $\mathcal{X}$,
$\tilde{S}, \tilde{S}', \tilde{S}_1, \tilde{S}'', \tilde{S}_2, \ldots$ to denote elements of $\tilde{\mathcal{S}}$,
$\tilde{X}, \tilde{X}', \tilde{X}_1, \tilde{X}'', \tilde{X}_2, \ldots$ to denote elements of $\tilde{\mathcal{X}}$,
$O, O', O_1, O'', O_2, \ldots$ to denote elements of $\mathcal{O}$.

The following definitions introduce the standard propositions of PEC+: v-propositions declare which value a fluent may possibly take, c-propositions model the causal relationships of a domain, i-propositions declare the initial conditions, p-propositions are used for the action occurrences, and h-propositions state that a given i-formula holds.

Definition 3.7 (v-Proposition)**.** A *v-proposition* has the form

$$F \textbf{ takes-values } \{V_1, \ldots, V_m\} \tag{3.1}$$

where $m \geq 1$ and $\{V_1, \ldots, V_m\} = vals(F)$.

Definition 3.8 (c-Proposition, Head and Body of a c-Proposition)**.** A *c-proposition* has the form

$$\theta \textbf{ causes-one-of } \{O_1, O_2, \ldots, O_m\} \tag{3.2}$$

where, for $i = 1, \ldots, m$, $O_i \in \mathcal{O}$, $\chi(O_i) \neq \chi(O_j)$ when $i \neq j$, θ is a formula such that[1] $\theta \models A = \top$ for at least one $A \in \mathcal{A}$, and $\pi(\{O_1, \ldots, O_m\}) = 1$. The formula *body*$(C) = \theta$ and the set *head*$(C) = \{O_1, \ldots, O_m\}$ are the *body* and *head* of C, respectively. Outcome O_i is often omitted from *head*(C) if $\chi(O_i) = \emptyset$ (leaving $\pi(O_i)$ implicit since $\pi(\{O_1, \ldots, O_m\}) = 1$).

Definition 3.9 (i-Proposition)**.** An *i-proposition* has the form

$$\textbf{initially-one-of } \{O_1, O_2, \ldots, O_m\} \tag{3.3}$$

where, for $i = 1, \ldots, m$, $O_i \in \mathcal{O}$, $\pi(\{O_1, \ldots, O_m\}) = 1$, $\chi(O_i) \in \tilde{\mathcal{S}}$, and $\chi(O_i) \neq \chi(O_j)$ when $i \neq j$.

Definition 3.10 (p-Proposition)**.** A *p-proposition* has the form

$$A \textbf{ performed-at } I \textbf{ with-prob } P^+ \textbf{ if-holds } \theta \tag{3.4}$$

where θ is a fluent formula, $P^+ \in (0, 1]$ and I is such that $I < I'$ for some other $I' \in \mathcal{I}$. When a p-proposition p has the form (3.4), then it is said that p *has instant* I.

$$A \textbf{ performed-at } I \textbf{ with-prob } P^+$$

is shorthand for the p-proposition

$$A \textbf{ performed-at } I \textbf{ with-prob } P^+ \textbf{ if-holds } \top$$

and

$$A \textbf{ performed-at } I \textbf{ if-holds } \theta$$

is shorthand for the p-proposition

$$A \textbf{ performed-at } I \textbf{ with-prob } 1 \textbf{ if-holds } \theta$$

[1] Here and in the following, literals of the language are to be interpreted as distinct propositions when using the standard propositional entailment $\models$.

and

$$A \textbf{ performed-at } I$$

is shorthand for the p-proposition

$$A \textbf{ performed-at } I \textbf{ with-prob } 1 \textbf{ if-holds } \top.$$

Notation 3.2. In the following, lowercase letters are generally used to denote propositions, e.g. $c, c', c_1, c'', c_2, \ldots$ will be used for c-propositions.

Definition 3.11 (Domain Description)**.** A *domain description* is a finite set $\mathcal{D}$ of v-propositions, c-propositions, p-propositions and i-propositions such that: (i) for any two distinct c-propositions in $\mathcal{D}$ with bodies θ and η, θ is incompatible with η (i.e., there is no state S such that $S \models \theta$ and $S \models \eta$), (ii) $\mathcal{D}$ contains exactly one i-proposition, (iii) $\mathcal{D}$ contains exactly one v-proposition for each $F \in \mathcal{F}$ and (iv) if a p-proposition "A **performed-at** I **with-prob** P^+ **if-holds** θ" belongs to $\mathcal{D}$, then for all $P' \in (0, 1]$ and formulas η that are compatible with θ (i.e. such that for some state S both $S \models \theta$ and $S \models \eta$ hold) there is no other p-proposition of the form "A **performed-at** I **with-prob** P' **if-holds** η" that belongs to $\mathcal{D}$.

Definition 3.12 (Action Narrative)**.** An *action narrative* is any finite set of p-propositions. For $\mathcal{D}$ a domain description, the action narrative *narr*($\mathcal{D}$) is the set of all p-propositions in $\mathcal{D}$.

Example 3.6 (Coin Toss Domain)**.** Scenario 1.5 can be modelled using the following domain description $\mathcal{D}_C$:

(C1) *Coin* **takes-values** $\{$*Heads*, *Tails*$\}$

(C2) **initially-one-of** $\{(\{$*Coin*$=$*Heads*$\}, 1)\}$

(C3) *Toss* **causes-one-of**
$\{(\{$*Coin*$=$*Heads*$\}, 0.49)$,
$(\{$*Coin* $=$ *Tails*$\}, 0.49)$,
$(\emptyset, 0.02)\}$

(C4) *Toss* **performed-at** 1

where *C1* is a v-proposition, *C2* is an i-proposition, *C3* is a c-proposition, *C4* is a p-proposition, and *narr*($\mathcal{D}_C$) $= \{$*C4*$\}$.

Example 3.7 (Antibiotic Domain)**.** Scenario 1.1 can be modelled using the following domain description $\mathcal{D}_A$:

(A1) *Bacteria* **takes-values** $\{$*Weak*, *Resistant*, *Absent*$\}$

(A2) *Rash* **takes-values** $\{$*Present*, *Absent*$\}$

(A3) **initially-one-of**
$\{(\{$*Bacteria*$=$*Weak*, *Rash*$=$*Present*$\}, 9/10)$,
$(\{$*Bacteria*$=$*Absent*, *Rash*$=$*Present*$\}, 1/10)\}$

(A4) *TakesMedicine* $\wedge$ *Bacteria* $=$ *Weak*
causes-one-of
$\{(\{Bacteria = Absent, Rash = Absent\}, 7/10),$
$(\{Bacteria = Resistant, Rash = Absent\}, 1/10),$
$(\{Bacteria = Resistant\}, 2/10)\}$

(A5) *TakesMedicine* $\wedge$ *Bacteria* $=$ *Resistant*
causes-one-of
$\{(\{Bacteria = Absent, Rash = Absent\}, 1/13),$
$(\emptyset, 12/13)\}$

(A6) *TakesMedicine* **performed-at** 1

(A7) *TakesMedicine* **performed-at** 3

where *A1* and *A2* are v-propositions, *A3* is an i-proposition, *A4* and *A5* are c-propositions, *A6* and *A7* are p-propositions, and $narr(\mathcal{D}_A) = \{A6, A7\}$.

Example 3.8 (Tuberculosis Domain)**.** Scenario 1.3 can be modelled using the following domain description $\mathcal{D}_T$:

(T1) *Tuberculosis* **takes-values** $\{Absent, Latent, Active\}$

(T2) **initially-one-of**
$\{(\{Tuberculosis = Active\}, 1/30),$
$(\{Tuberculosis = Latent\}, 9/30),$
$(\{Tuberculosis = Absent\}, 2/3)\}$

(T3) *Exposure* $\wedge$ *Tuberculosis* $=$ *Absent*
causes-one-of
$\{(\{Tuberculosis = Active\}, 4/100),$
$(\{Tuberculosis = Latent\}, 76/100),$
$(\emptyset, 2/10)\}$

(T4) *Reactivation* $\wedge$ *Tuberculosis* $=$ *Latent*
causes-one-of
$\{(\{Tuberculosis = Active\}, 1)\}$

(T5) *Reactivation* **performed-at** I **with-prob** $8/10^4$
if-holds *Tuberculosis* $=$ *Latent*

(T6) *Exposure* **performed-at** 0 **with-prob** $25/100$

(T7) *Exposure* **performed-at** 2 **with-prob** $9/10$

Given the convention that instants represent years from the beginning of the story told by the patient, and since the first reported exposure was at age 30, the p-proposition *T6* refers to an exposure when the patient was 30 and the p-proposition *T7* refers to an exposure at patient's age 32.

Notice that *T5* is a *proposition scheme*, i.e. it specifies a set of p-propositions, each of which is obtained from *T5* by replacing I with an instant in $\mathcal{I}_T$. Notice that since $\mathcal{I}_T$ is finite, this is consistent with definition 3.11 which requires the set of propositions to be finite.

Numeric data in this example has been adapted from [1] (see also fig. 1.1) but some significant and possibly unrealistic simplifications have been made, namely: the probability of infection upon exposure has been fixed to 80%, the initial probability of not having TB has been assumed to follow the proportion $2:1$ (based on the fact that $\approx 2/3$ of the population are not infected). It has also been assumed that only 10% of those who are initially infected have active TB (based on the fact that $\approx 10\%$ of people who are infected with TB develop active TB during lifetime), and that people with latent TB have a 0.08% yearly probability of reactivation.

However, it should be noted that accurate data can be readily introduced in this model and that PEC+ can account for many other sophistications. For example, the chance of reactivation is known to be higher within 7 years of infection: this can be modelled through the use of a counter and an appropriate set of p-propositions; when patients are exposed but remain uninfected, they develop resistance to the pathogen, and become less vulnerable to it: this can be modelled using a similar mechanism to that of example 3.7; finally, HIV-positive patients with latent TB have a $\approx 10\%$ yearly chance of reactivation, which is much higher than the assumed $\approx 5\%$ lifetime risk of reactivation: this can be modelled by introducing a fluent indicating HIV infection and conditioning p-propositions on it.

Finally, h-propositions are entailed by domain descriptions:

Definition 3.13 (h-proposition, conditional h-proposition)**.** An *h-proposition* has the form

$$\varphi \textbf{ holds-with-prob } P \tag{3.5}$$

A *conditional h-proposition* has the form

$$\textbf{given } \psi, \varphi \textbf{ holds-with-prob } P \tag{3.6}$$

for i-formulas φ and ψ.

For example, the following sections show the formal sense in which $\mathcal{D}_C$ entails the h-proposition $[\mathit{Coin}=\mathit{Heads}]@2$ **holds-with-prob** 0.51.

3.2 Semantics

For the remainder of this thesis, $\mathcal{D}$ is an arbitrary domain description.

Definition 3.14 (Worlds)**.** A *world* is a function $W : \mathcal{I} \to \mathcal{S}$. The set of all worlds is denoted by $\mathcal{W}$.

Notation 3.3. In the following, $W, W', W'', W_1, W_2, \ldots$ denote worlds.

Definition 3.15 (Satisfaction of an i-formula, Logical Consequence for i-formulas)**.** Given a world W and a literal L, W *satisfies an i-formula* $[L]@I$, written $W \mid\!\models [L]@I$, iff $L \in W(I)$[2].

[2]The symbols $\mid\!\models$ and $\mid\!\not\models$ should not be confused with $\models$ and $\not\models$ which stand for classical propositional entailment

Otherwise, $W \not\models [L]@I$. The definition of $\models$ is recursively extended for arbitrary i-formulas as follows: if φ and ψ are i-formulas, $W \models \varphi \wedge \psi$ iff $W \models \varphi$ and $W \models \psi$, and $W \models \neg\varphi$ iff $W \not\models \varphi$. Connectives $\vee$ and $\rightarrow$ are defined as shorthand in the usual way, i.e. $\varphi \vee \psi$ is shorthand for $\neg(\neg\varphi \wedge \neg\psi)$, and $\varphi \rightarrow \psi$ is shorthand for $\neg(\varphi \wedge \neg\psi)$. Given a (possibly empty) set Δ of i-formulas, $W \models \Delta$ iff $W \models \psi$ for all $\psi \in \Delta$. Given an i-formula φ and a set Δ of i-formulas $\Delta \models \varphi$ if for all $W \in \mathcal{W}$ such that $W \models \Delta$, $W \models \varphi$ also holds. For two i-formulas φ and ψ, $\psi \models \varphi$ is shorthand for $\{\psi\} \models \varphi$, and $\models \varphi$ is shorthand for $\emptyset \models \varphi$.

Example 3.9. Three worlds for Scenario 1 can be specified as follows:

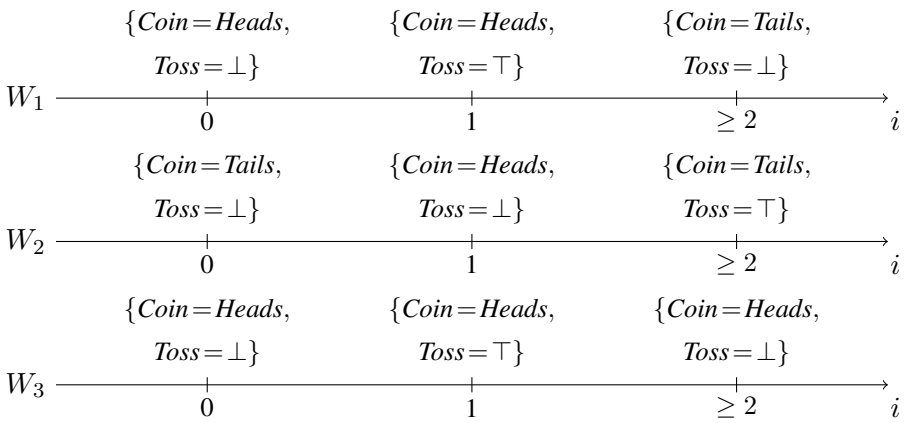
$W_1(0) = \{\mathit{Coin}=\mathit{Heads},\ \mathit{Toss}=\bot\}$,
$W_1(1) = \{\mathit{Coin}=\mathit{Heads},\ \mathit{Toss}=\top\}$,
$W_1(I) = \{\mathit{Coin}=\mathit{Tails},\ \mathit{Toss}=\bot\}$ for all $I \geq 2$.

$W_2(0) = \{\mathit{Coin}=\mathit{Tails},\ \mathit{Toss}=\bot\}$,
$W_2(1) = \{\mathit{Coin}=\mathit{Heads},\ \mathit{Toss}=\bot\}$,
$W_2(I) = \{\mathit{Coin}=\mathit{Tails},\ \mathit{Toss}=\top\}$ for all $I \geq 2$.

$W_3(0) = \{\mathit{Coin}=\mathit{Heads},\ \mathit{Toss}=\bot\}$,
$W_3(1) = \{\mathit{Coin}=\mathit{Heads},\ \mathit{Toss}=\top\}$,
$W_3(I) = \{\mathit{Coin}=\mathit{Heads},\ \mathit{Toss}=\bot\}$ for all $I \geq 2$.

Intuitively, W_1 and W_3 match the domain description in example 3.6 as they represent a coherent history of what could have happened in scenario 1.5, whereas W_2 does not (e.g., changes occur when no action is performed, an infinite number of actions is being performed, etc...). This intuition will be made precise in what follows.

Since worlds are functions from instants to states, they can conveniently be depicted as timelines:



Definition 3.16 (Closed World Assumption for Actions)**.** A world W is said to satisfy the *closed world assumption for actions* (or *CWA for actions*, for short) *w.r.t.* $\mathcal{D}$ if it satisfies the following conditions: (i) for all $A \in \mathcal{A}$ and $I \in \mathcal{I}$, if $W \models [A]@I$ then there exists some $P^+ \in (0, 1]$ and a fluent formula θ such that $W \models [\theta]@I$ and

"A **performed-at** I **with-prob** P^+ **if-holds** θ" is in $\mathcal{D}$, (ii) if for some $A \in \mathcal{A}$, $I \in \mathcal{I}$ and formula θ the p-proposition "A **performed-at** I **with-prob** 1 **if-holds** θ" is in $\mathcal{D}$ and $W \mid\!\models [\theta]@I$ then it must be the case that $W \mid\!\models [A]@I$.

Example 3.10. Let W_1, W_2 and W_3 be the worlds in example 3.9, and let $\mathcal{D}_C$ be the domain description in example 3.6. World W_1 satisfies CWA for actions w.r.t. $\mathcal{D}_C$ as *Toss* $\in W_1(I)$ if and only if $I = 1$, which is consistent with *C4* being the only p-proposition in $\mathcal{D}_C$. CWA is not satisfied by W_2 as $\neg$*Toss* $\in W_2(1)$, i.e. $W_2 \mid\!\models [\neg \textit{Toss}]@1$, but this is not consistent with *C4*. W_3 satisfies CWA for actions for the same reason as W_1.

Example 3.11. Consider $\mathcal{D}_T$ as in example 3.8 and consider the following worlds:

$W'(I) = \{\textit{Tuberculosis}=\textit{Absent},\ \textit{Exposure}=\bot, \textit{Reactivation}=\bot\}$ for all $I \in \mathcal{I}_T$

$W''(0) = \{\textit{Tuberculosis}=\textit{Absent},\ \textit{Exposure}=\top, \textit{Reactivation}=\bot\}$
$W''(1) = \{\textit{Tuberculosis}=\textit{Active},\ \textit{Exposure}=\bot, \textit{Reactivation}=\top\}$
$W''(I) = \{\textit{Tuberculosis}=\textit{Active},\ \textit{Exposure}=\bot, \textit{Reactivation}=\bot\}$ for $2 \leq I \leq 50$

World W' satisfies CWA for actions as $W' \mid\!\models [\neg A]@I$ for all $A \in \mathcal{A}_T$ and $I \in \mathcal{I}_T$ (which satisfies requirement (i) of definition 3.16) and since no p-proposition of the form "A **performed-at** I **with-prob** 1 **if-holds** θ" belongs to $\mathcal{D}_T$ (which satisfies requirement (ii)). However, world W'' does not satisfy CWA for actions as $W'' \mid\!\models [\textit{Reactivation}=\top]@1$ but there is no p-proposition "*Reactivation* **performed-at** 1 **with-prob** P^+ **if-holds** θ" in $\mathcal{D}_T$ such that $W'' \mid\!\models [\theta]@1$. In fact, $\mathcal{D}_T$ contains a p-proposition "*Reactivation* **performed-at** 1 **with-prob** $8/10^4$ **if-holds** *Tuberculosis*$=$*Latent*", but it is not the case that $W'' \mid\!\models [\textit{Tuberculosis}=\textit{Latent}]@1$.

Definition 3.17 (Cause Occurrence)**.** Let θ be the body of a c-proposition c in a domain description $\mathcal{D}$ and $I \in \mathcal{I}$. If $W \mid\!\models [\theta]@I$ then it is said that *a cause occurs at instant I in W w.r.t. to $\mathcal{D}$*, and that *the c-proposition c is activated at I in W w.r.t. $\mathcal{D}$*. The set $\textit{occ}_{\mathcal{D}}(W)$ is the set $\{I \in \mathcal{I} \mid \text{a cause occurs at } I \text{ in } W\}$. The function $\textit{cprop}_{\mathcal{D}}$ with domain $\{(W, I) \mid W \in \mathcal{W}, I \in \textit{occ}_{\mathcal{D}}(W)\}$ is defined for instants I in its domain as $\textit{cprop}_{\mathcal{D}}(W, I) = c$ where c is the (unique) c-proposition activated at I in world W.

Example 3.12. Let $\mathcal{D}_C$ be as in example 3.6 and W_1, W_2 and W_3 be as in example 3.9. Since $W_1 \mid\!\models [\textit{Toss}]@I$ if and only if $I = 1$ (and similarly for W_3), it follows that $\textit{occ}_{\mathcal{D}_C}(W_1) = \textit{occ}_{\mathcal{D}_C}(W_3) = \{1\}$, with $\textit{cprop}_{\mathcal{D}_C}(W_1, 1) = \textit{cprop}_{\mathcal{D}_C}(W_3, 1) = \textit{C3}$. For W_2, $\textit{occ}_{\mathcal{D}_C}(W_2)$ is defined as $\{I \mid I \in \mathbb{N}, I \geq 2\}$ with $\textit{cprop}_{\mathcal{D}_C}(W_2, I) = \textit{C3}$ for $I \geq 2$.

Definition 3.18 (Initial Choice)**.** Let $\mathcal{D}$ be a domain description and the unique i-proposition in $\mathcal{D}$ be of the form (3.3). Each $O_1, O_2, \ldots, O_m$ is called an *initial choice w.r.t. $\mathcal{D}$*.

Definition 3.19 (Effect Choice)**.** Let W be a world and $\mathcal{D}$ a domain description. An *effect choice for W w.r.t. $\mathcal{D}$* is a function $ec : \textit{occ}_{\mathcal{D}}(W) \rightarrow \mathcal{O}$ such that for all instants $I \in \textit{occ}_{\mathcal{D}}(W)$, $ec(I) \in \textit{head}(\textit{cprop}_{\mathcal{D}}(W, I))$.

Example 3.13. Let $\mathcal{D}_C$ be as in example 3.6 and W_1, W_2 and W_3 be as in example 3.9. The only initial choice w.r.t. $\mathcal{D}_C$ is $ic_1 = (\{\textit{Coin}{=}\textit{Heads}\}, 1)$. The only effect choices for W_1 w.r.t. $\mathcal{D}_C$ are $ec_1(1) = (\{\textit{Coin}{=}\textit{Tails}\}, 49/100)$, $ec_2(1) = (\{\textit{Coin}{=}\textit{Heads}\}, 49/100)$ and $ec_3(1) = (\emptyset, 2/100)$. Notice that since $occ_{\mathcal{D}_C}(W_1) = occ_{\mathcal{D}_C}(W_3)$, all the effect choices for W_1 are also effect choices for W_3. There are an (uncountably) infinite number of effect choices for W_2 w.r.t. $\mathcal{D}_C$, each one mapping each instant $I \geq 2$ to $(\{\textit{Coin}{=}\textit{Heads}\}, 49/100)$, $(\{\textit{Coin}{=}\textit{Tails}\}, 49/100)$ or $(\emptyset, 2/100)$.

Definition 3.20 (Initial Condition)**.** A world W is said to *satisfy the initial condition w.r.t.* $\mathcal{D}$ if there exists an initial choice *ic* w.r.t. $\mathcal{D}$ such that $W(\bar{0})\restriction\mathcal{F} = \chi(ic)$. If a world W satisfies the initial condition w.r.t. $\mathcal{D}$ for some initial choice *ic*, then it is said that W *and ic are consistent with each other w.r.t.* $\mathcal{D}$.

Example 3.14. Let $\mathcal{D}_C$ be as in example 3.6, and W_1, W_2 and W_3 be as in example 3.9. Since $ic_1 = (\{\textit{Coin}{=}\textit{Heads}\}, 1)$ is the only initial choice w.r.t. $\mathcal{D}_C$ as outlined in example 3.13, W_1 and W_3 are consistent w.r.t. $\mathcal{D}_C$ with it, since $W_1(0)\restriction\mathcal{F} = W_3(0)\restriction\mathcal{F} = \chi(ic_1)$. Therefore, W_1 and W_3 satisfy the initial Condition w.r.t. $\mathcal{D}_C$. Since $W_2(0)\restriction\mathcal{F} \neq \chi(ic_1)$, W_2 does not satisfy the initial Condition.

Definition 3.21 (Intervals)**.** Given two instants I and I' such that $I \leq I'$, the intervals $[I, I']$, $[I, I')$, $(I, I']$ and (I, I') are defined in the standard way w.r.t. the total order $\leq$. Also, $[I, +\infty)$ is shorthand for the set $\{I' \mid I' \in \mathcal{I}, I' \geq I\}$, $(-\infty, I]$ as shorthand for $\{I' \mid I' \in \mathcal{I}, I' \leq I\}$, $(I, +\infty)$ as a shorthand for $[I, \infty) \setminus \{I\}$ and $(-\infty, I)$ as a shorthand for $(-\infty, I] \setminus \{I\}$.

Definition 3.22 (Fluent State Update)**.** Given a fluent state $\tilde{S}$ and a partial fluent state $\tilde{X}$, the *update of* $\tilde{S}$ *w.r.t.* $\tilde{X}$, written $\tilde{S} \oplus \tilde{X}$, is the fluent state $(\tilde{S} \ominus \tilde{X}) \cup \tilde{X}$, where $\tilde{S} \ominus \tilde{X}$ is the partial fluent state formed by removing all fluent literals from $\tilde{S}$ of the form $F = V$ for some F and V' such that $F = V' \in \tilde{X}$. The operator $\oplus$ is left-associative, so e.g. $\tilde{S} \oplus \tilde{X} \oplus \tilde{X}'$ is understood as $((\tilde{S} \oplus \tilde{X}) \oplus \tilde{X}')$.

Definition 3.23 (Justified Change)**.** A world W is said to satisfy the *justified change condition w.r.t.* $\mathcal{D}$ if and only if there exists an effect choice *ec* w.r.t. $\mathcal{D}$ such that for all instants I and I' with $I < I'$, *ec* maps the possibly empty set of instants in $occ_{\mathcal{D}}(W) \cap [I, I') = \{I_1, \ldots, I_n\}$ to $O_1, O_2, \ldots, O_n$ respectively, where $I_1, \ldots, I_n$ are ordered w.r.t. $\leq$, and

$$W(I')\restriction\mathcal{F} = (W(I)\restriction\mathcal{F}) \oplus \chi(O_1) \oplus \chi(O_2) \oplus \cdots \oplus \chi(O_n) \tag{3.7}$$

If a world W satisfies the justified change condition for some effect choice *ec*, W and *ec* are said to be *consistent* with each other *w.r.t.* $\mathcal{D}$.

Example 3.15. Let $\mathcal{D}_C$ be as in example 3.6, W_1, W_2 be as in example 3.9, and ec_1 be defined as in example 3.13.

For any two instants I, $I' \in \mathbb{N}$ with $I < I'$, if $[I, I') \cap occ_{\mathcal{D}_C}(W_1) = \emptyset$ then clearly $W_1(I')\restriction\mathcal{F} = W_1(I)\restriction\mathcal{F}$. Otherwise, if $[I, I') \cap occ_{\mathcal{D}_C}(W_1) \neq \emptyset$, i.e. $[I, I') \cap occ_{\mathcal{D}_C}(W_1) = \{1\}$ then (3.7) holds as $W_1(I)\restriction\mathcal{F} \oplus \chi(ec_1(1)) = \{\textit{Coin}{=}\textit{Tails}\} = W(I')\restriction\mathcal{F}$. So the Justified Change Condition w.r.t. $\mathcal{D}_C$ is satisfied by W_1.

For W_2 to satisfy the Justified Change Condition w.r.t. $\mathcal{D}_C$, equation (3.7) would require $W_2(0)\restriction\mathcal{F} = W_2(1)\restriction\mathcal{F}$ (as $occ_{\mathcal{D}_C}(W_2) \cap [1,2) = \emptyset$), but this is not the case. Hence, W_2 does not satisfy the Justified Change Condition w.r.t. $\mathcal{D}_C$.

Definition 3.24 (Well-behaved Worlds)**.** A world is said to be *well-behaved w.r.t.* $\mathcal{D}$ if it satisfies CWA for actions, the initial condition and the Justified Change Condition w.r.t. $\mathcal{D}$. The set of well-behaved worlds w.r.t. $\mathcal{D}$ is denoted by $\mathcal{W}_{\mathcal{D}}$.

Example 3.16. Let $\mathcal{D}_C$ be as in example 3.6 and W_1, W_2 be as in example 3.9. W_1 is well-behaved as it satisfies CWA for actions (see example 3.10), the initial condition (see example 3.14) and the Justified Change Condition (see example 3.15) w.r.t. $\mathcal{D}_C$. W_2 is not well-behaved as it fails to satisfy any of these conditions.

Definition 3.25 ($\mathcal{D}$-entailment)**.** Given an i-formula φ and a set Δ of i-formulas, $\Delta \mid\mid\!\models_{\mathcal{D}} \varphi$ if for all $W \in \mathcal{W}_{\mathcal{D}}$ such that $W \mid\mid\!\models \Delta$, $W \mid\mid\!\models \varphi$ also holds. For two i-formulas ψ and φ, $\psi \mid\mid\!\models_{\mathcal{D}} \varphi$ is shorthand for $\{\psi\} \mid\mid\!\models_{\mathcal{D}} \varphi$, and $\mid\mid\!\models_{\mathcal{D}} \varphi$ is shorthand for $\emptyset \mid\mid\!\models_{\mathcal{D}} \varphi$.

In the following two definitions, the symbol $\boxtimes$ is used to represent the period or point in time "just before" the least instant $\bar{0}$, and therefore "just before" any action can occur. Intuitively, a trace represents a "causal justification" for a world.

Definition 3.26 (Candidate Trace)**.** A *candidate trace* is a function $tr : dom(tr) \cup \{\boxtimes\} \to \mathcal{O}$ where $dom(tr) \subseteq \mathcal{I}$ and $\boxtimes$ is a new symbol such that $\boxtimes \notin \mathcal{I}$. For readability, when $dom(tr)$ is finite, tr is sometimes written in the form

$$\langle tr(\boxtimes)@\boxtimes, tr(I_1)@I_1, \ldots, tr(I_m)@I_m\rangle$$

where $dom(tr) = \{I_1, \ldots, I_m\}$ and instants are ordered w.r.t. $\leq$. The shorthand $dom^+(tr)$ stands for $dom(tr) \cup \{\boxtimes\}$.

Definition 3.27 (Trace)**.** If W is a well-behaved world w.r.t. $\mathcal{D}$, then a candidate trace tr is a *trace of* W *w.r.t.* $\mathcal{D}$ if there exist an initial choice *ic* and an effect choice *ec* consistent with W such that:

$$tr = \langle ic@\boxtimes, ec(I_1)@I_1, \ldots, ec(I_m)\rangle \tag{3.8}$$

where $\{I_1, \ldots, I_m\} = occ_{\mathcal{D}}(W)$. In this case, expression (3.8) can be shortened to $tr = \langle ic, ec\rangle$.

For any $W \in \mathcal{W}$, $\mathcal{TR}_{\mathcal{D}}^{W}$ stands for the set of all traces of W w.r.t. $\mathcal{D}$, and if W is not well-behaved $\mathcal{TR}_{\mathcal{D}}^{W} = \emptyset$.

A well-behaved world can have multiple traces (i.e. multiple ways the world could have been "caused"), as shown in the following example.

Example 3.17. Let $\mathcal{D}_C$ be as in example 3.6, W_3 be as in example 3.9 and ic_1, ec_2, ec_3 be as defined in example 3.13. World W_3 has two distinct traces, $tr_3' = \langle ic_1, ec_2\rangle$ and $tr_3'' = \langle ic_1, ec_3\rangle$, which disagree on the effect choice: in one case the robot manages to toss the coin producing *Coin=Heads* as a result (i.e., $tr_3'(1) = (\{Coin{=}Heads\}, 0.49)$) whereas in the other case the robot fails to grab the coin (i.e., $tr_3''(1) = (\emptyset, 0.02)$) leaving *Coin=Heads* to hold. These two traces are also the only traces of this world w.r.t. $\mathcal{D}_C$.

However, for some candidate traces tr there exists no well-behaved world W such that tr is a trace of W, e.g. there is no well-behaved world w.r.t. $\mathcal{D}_C$ as in example 3.6 having trace $\langle(\{\mathit{Coin}=\mathit{Tails}\},1)@\boxtimes\rangle$ as $(\{\mathit{Coin}=\mathit{Tails}\},1)$ is not an initial choice for $\mathcal{D}_C$.

Definition 3.27 is now generalised to domain descriptions:

Definition 3.28 (Trace of a Domain Description)**.** Given a candidate trace tr, if there exists a (well-behaved) world W such that $tr \in \mathcal{TR}^W_{\mathcal{D}}$, then tr is said to be *a trace of* $\mathcal{D}$. For a trace tr of $\mathcal{D}$, W_{tr} denotes the corresponding well-behaved world.

Definition 3.29 (Evaluation of a Trace)**.** Let tr be a candidate trace. The *evaluation of* tr, written $\epsilon(tr)$, is defined as:

$$\epsilon(tr) = \prod_{I \in \mathit{dom}^+(tr)} \pi(tr(I)) \tag{3.9}$$

Definition 3.30 (Evaluation of a Narrative)**.** Given a p-proposition p of the form "A **performed-at** I **with-prob** P^+ **if-holds** θ", the *evaluation of* p *w.r.t.* W is defined as

$$\epsilon(p,W) = \begin{cases} 1 & \text{if } W \mid\!\not\models [\theta]@I \\ P^+ & \text{if } W \mid\!\models [\theta]@I \text{ and } W \mid\!\models [A]@I \\ 1-P^+ & \text{if } W \mid\!\models [\theta]@I \text{ and } W \mid\!\models [\neg A]@I \end{cases} \tag{3.10}$$

For an action narrative N (see definition 3.12) the previous definition is extended to:

$$\epsilon(N,W) = \prod_{p \in N} \epsilon(p,W). \tag{3.11}$$

and $\epsilon_{\mathcal{D}}(W)$ is shorthand for $\epsilon(\mathit{narr}(\mathcal{D}),W)$. Conventionally, $\epsilon(\emptyset,W)=1$ for all worlds W.

Notice that, from eqs. (3.10) and (3.11), it follows that p-propositions are only dependent on the current state of the environment (see θ in eq. (3.10)), and that p-propositions of the form "A **performed-at** I" (i.e. when $\theta = \top$ in eq. (3.10)) are independent from each other.

Example 3.18. Recall world W' from example 3.11. Since $W' \mid\!\not\models [\mathit{Tuberculosis}=\mathit{Latent}]@I$ for all instants $I \in \mathcal{I}_T$, the precondition of axiom schema *T5* is never satisfied, hence $\epsilon(p,W')=1$ for all instances p of *T5*. Since $W' \mid\!\models [\mathit{Exposure}=\bot]@0$, $\epsilon(\mathit{T6},W') = 1-25/100 = 75/100$ and since $W' \mid\!\models [\mathit{Exposure}=\bot]@2$, $\epsilon(\mathit{T7},W') = 1-9/10 = 1/10$. Therefore, $\epsilon_{\mathcal{D}_T}(W') = 75/100 \cdot 1/10 = 75/1000$.

The two definitions that follow introduce functions that assign a numerical weight to each world $W \in \mathcal{W}$ which is taken to represent the degree of plausibility of that world. While definition 3.31 defines the set of all such possible functions, definition 3.32 defines a specific function, called a *model*, which will later on be shown to satisfy the axioms of probability.

Definition 3.31 ($[0,1]$-interpretation)**.** A *[0,1]-interpretation* is a function from $\mathcal{W}$ to $[0,1]$.

Definition 3.32 (Model)**.** A *model* of a domain description $\mathcal{D}$ is a $[0,1]$-interpretation $M_{\mathcal{D}}$ such that

1. If $W \in \mathcal{W}$ is not well-behaved w.r.t. $\mathcal{D}$,

$$M_{\mathcal{D}}(W) = 0 \tag{3.12}$$

2. If $W \in \mathcal{W}$ is well-behaved w.r.t. $\mathcal{D}$,

$$M_{\mathcal{D}}(W) = \epsilon_{\mathcal{D}}(W) \cdot \sum_{tr \in \mathcal{TR}_{\mathcal{D}}^{W}} \epsilon(tr) \tag{3.13}$$

Example 3.19. Let $\mathcal{D}_C$ be as in example 3.6 and W_3 be as example 3.9. As discussed in example 3.17, W_3 has exactly two traces $tr_3' = \langle ic_1, ec_2 \rangle$ and $tr_3'' = \langle ic_1, ec_3 \rangle$. Equations (3.9) and (3.13) yield:

$$M_{\mathcal{D}_C}(W_3) = \epsilon(tr_3') + \epsilon(tr_3'') = 0.49 + 0.02 = 0.51$$

Similarly, W_1 as in example 3.9 has a unique trace $\langle(\{\mathit{Coin}{=}\mathit{Heads}\}, 1)@\boxtimes, (\{\mathit{Coin}{=}\mathit{Tails}\}, 0.49)@1\rangle$, and therefore $M_{\mathcal{D}_C}(W_1) = 0.49$.

Proposition 3.1. A domain description $\mathcal{D}$ has a unique model.

Proof. This can be derived from definition 3.32 by considering that $M_{\mathcal{D}}(W)$ is calculated as a product of functions of the states of W. □

Definition 3.33. The model $M_{\mathcal{D}}$ is extended to a function $M_{\mathcal{D}} : \Phi \to [0, 1]$ over i-formulas in the following way:

$$M_{\mathcal{D}}(\varphi) = \sum_{W \mid\models \varphi} M_{\mathcal{D}}(W) \tag{3.14}$$

If ψ is an i-formula such that $M_{\mathcal{D}}(\psi) \neq 0$, then the function $M_{\mathcal{D}}(\cdot \mid \psi) : \Phi \to [0, 1]$ is defined as follows:

$$M_{\mathcal{D}}(\varphi \mid \psi) = \frac{M_{\mathcal{D}}(\varphi \wedge \psi)}{M_{\mathcal{D}}(\psi)}$$

Definition 3.34 (Entailment for Domain Descriptions)**.** Given a domain description $\mathcal{D}$ and two i-formulas φ and ψ, the h-proposition "φ **holds-with-prob** P" is *entailed by* $\mathcal{D}$ iff $M_{\mathcal{D}}(\varphi) = P$. This can be written

$$\mathcal{D} \mid\models \varphi \textbf{ holds-with-prob } P$$

The conditional h-proposition "**given** ψ, φ **holds-with-prob** P" is *entailed by* $\mathcal{D}$ iff $M_{\mathcal{D}}(\varphi \mid \psi) = P$ and write

$$(\mathcal{D} \mid \psi) \mid\models \varphi \textbf{ holds-with-prob } P$$

or equivalently

$$\mathcal{D} \mid\models \textbf{given } \psi, \varphi \textbf{ holds-with-prob } P$$

Example 3.20. For $\mathcal{D}_C$ as in example 3.6, the only well-behaved world W such that $W \mid\models [\mathit{Coin}{=}\mathit{Heads}]@2$ is W_3. Definition 3.33 and example 3.19 yield

$$M_{\mathcal{D}_C}([\mathit{Coin}{=}\mathit{Heads}]@2) = M_{\mathcal{D}_C}(W_3) = 0.51.$$

Similarly, $[\mathit{Coin}=\mathit{Heads}]@0$ yields

$$M_{\mathcal{D}_C}([\mathit{Coin}=\mathit{Heads}]@0) = M_{\mathcal{D}_C}(W_1) + M_{\mathcal{D}_C}(W_3) = 1$$

and from this it follows that $\mathcal{D}_C$ entails the two following h-propositions:

$[\mathit{Coin}=\mathit{Heads}]@2$ **holds-with-prob** 0.51,

$[\mathit{Coin}=\mathit{Heads}]@0$ **holds-with-prob** 1.

3.3 Properties of a model

As shown in section 3.2, worlds are possible histories of what could have happened in a given scenario modelled by an appropriate domain description. Notice also that worlds are mutually exclusive, in the sense that if one of them is believed to be the one representing the "true" history, then the others cannot be believed to be also true. Therefore, given a domain $\mathcal{D}$, belief in a given world can be represented by a random variable (see section 2.2.1) with domain the set $\mathcal{W}_\mathcal{D}$ of well-behaved worlds w.r.t. $\mathcal{D}$. The remainder of this section aims at proving that the model $M_\mathcal{D}$ is indeed a probability distribution satisfying constraint (2.3).

Furthermore, consider the following adaptation of definition 2.3 to the present context:

Definition 3.35 (Probability Function over i-Formulas)**.** A *probability function over i-formulas* is a function $p : \Phi \to [0, 1]$ such that:

1. if $||\!=\varphi$, then $p(\varphi) = 1$,
2. if $\varphi \;||\!=\neg\psi$ for two i-formulas φ and ψ, then $p(\varphi \vee \psi) = p(\varphi) + p(\psi)$.

The associated *conditional probability* of φ given ψ is defined as

$$p(\varphi \mid \psi) = \frac{p(\varphi \wedge \psi)}{p(\psi)} \tag{3.15}$$

for $p(\psi) \neq 0$.

In the remainder of this section it is proved that both $M_\mathcal{D}$ and $M_\mathcal{D}(\cdot \mid \psi)$ are probability functions.

In order to do so, some auxiliary definitions and prove intermediate results are introduced first:

Definition 3.36 (Restricted Domain Description)**.** If $\mathcal{D}$ is a domain description, $\mathcal{D}_{\leq I}$ denotes the domain description obtained from $\mathcal{D}$ by removing all the p-propositions occurring at instants $> I$, and similarly $\mathcal{D}_{<I}$ denotes the domain description obtained from $\mathcal{D}$ by removing all the p-propositions occurring at instants $\geq I$. Finally, $\mathcal{D}_\emptyset$ denotes the domain description obtained from $\mathcal{D}$ by removing all p-propositions, i.e. $\mathcal{D}_{<\bar{0}}$.

Definition 3.37 (Fluent-indistinguishability, Indistinguishability)**.** A world W is said to be *fluent-indistinguishable from W' up to an instant I* if and only if $W(I')\restriction\mathcal{F} = W'(I')\restriction\mathcal{F}$ for all instants I' such that $I' \leq I$. W is said to be *indistinguishable from W' up to an instant I* if and only if it is fluent indistinguishable from W' up to I and if for all $I' < I$ it also satisfies $A \in W(I')$ if and only if $A \in W'(I')$.

The following example illustrates the two concepts of restricted domain description and indistinguishability:

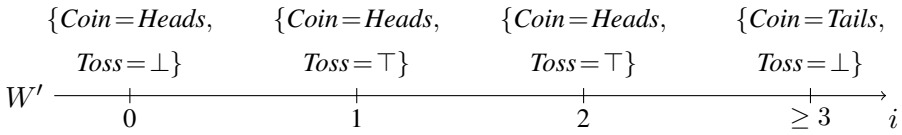
Example 3.21. Let $\mathcal{D}'$ be the domain description obtained from $\mathcal{D}_C$ as in example 3.6 by adding the following p-proposition:

(C5) *Toss* **performed-at** 2

and consider the following well-behaved world w.r.t. $\mathcal{D}'$:

$W'(0) = \{\mathit{Coin}{=}\mathit{Heads}, \neg\mathit{Toss}\}$,
$W'(1) = W'(2) = \{\mathit{Coin}{=}\mathit{Heads}, \mathit{Toss}\}$,
$W'(I) = \{\mathit{Coin}{=}\mathit{Tails}, \neg\mathit{Toss}\}$ for all $I > 2$

that can be visualised as follows:



W' has exactly two traces:

$$\langle(\{\mathit{Coin}{=}\mathit{Heads}\}, 1)@\mathbb{X}, (\{\mathit{Coin}{=}\mathit{Heads}\}, 0.49)@1, (\{\mathit{Coin}{=}\mathit{Tails}\}, 0.49)@2\rangle$$

and

$$\langle(\{\mathit{Coin}{=}\mathit{Heads}\}, 1)@\mathbb{X}, (\emptyset, 0.02)@1, (\{\mathit{Coin}{=}\mathit{Tails}\}, 0.49)@2\rangle$$

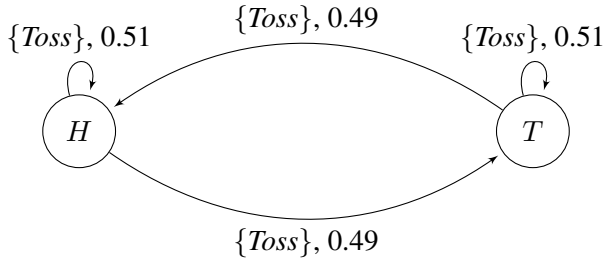
Consider $\mathcal{D}'_{<2}$ and notice that it coincides with $\mathcal{D}_C$. There is a unique well-behaved world w.r.t. $\mathcal{D}_C$ that is indistinguishable from W' up to 2, and this world is W_3 as in example 3.9.

Definition 3.38 (Transition Set, Transition Function)**.** Given a domain description $\mathcal{D}$, a state S and a fluent state $\tilde{S}'$, the *transition set* $\mathit{tset}_{\mathcal{D}}(S, \tilde{S}')$ is defined as follows: if $\mathcal{D}$ contains a (unique) c-proposition c such that S entails $\mathit{body}(c)$, then $\mathit{tset}_{\mathcal{D}}(S, \tilde{S}') = \{O \in \mathit{head}(c) \mid (S\restriction\mathcal{F}) \oplus \chi(O) = \tilde{S}'\}$ if there is no such c-proposition and $S\restriction\mathcal{F} = \tilde{S}'$ then $\mathit{tset}(S, \tilde{S}') = \{(\emptyset, 1)\}$; otherwise, $\mathit{tset}_{\mathcal{D}}(S, \tilde{S}') = \emptyset$.

The *transition function* for a domain description $\mathcal{D}$ is the function $t_{\mathcal{D}} : \mathcal{S} \times \tilde{\mathcal{S}} \to [0, 1]$ defined by $t_{\mathcal{D}}(S, \tilde{S}') = \pi(\mathit{tset}_{\mathcal{D}}(S, \tilde{S}'))$ (recall definition 3.6 for the meaning of π).

Informally, the transition function gives the probability of moving from state S to the fluent state $\tilde{S}'$ within $\mathcal{D}$, independently of its particular narrative.

The transition function for example 3.6 can be visualised as follows:



where the nodes represent fluent states (in this case the two nodes H and T stand for the fluent states $\{Coin{=}Heads\}$ and $\{Coin{=}Tails\}$ respectively), and if $p = t_{\mathcal{D}}(S, \tilde{S}')$ for some state S and some fluent state $\tilde{S}'$, then there is an arrow from a node representing $S{\restriction}\mathcal{F}$ to a node representing $\tilde{S}'$ which is labelled $S{\restriction}\mathcal{A}, p$. The arrow is omitted in some trivial cases (for instance when the set of actions is empty). Similarly, the transition function for example 3.7 can be pictured as in fig. 3.1.

At this point it is worth recalling that the model of a domain description is a function over *worlds* (i.e., entire time-lines). The transition function, on the other hand, describes how the future of an agent in a particular state of the world can *branch* according to the specific action being performed. The next proposition uses the transition function to illustrate the relationship between the narrative and causal aspect of a domain by relating those worlds whose evolution matches each others' up to some instant and then diverges due to the execution of different actions. This is done by using the transition function to conveniently express $M_{\mathcal{D}}(W)$ in terms of the model of a well-behaved world w.r.t. an appropriately restricted domain description:

Proposition 3.2. Let $\mathcal{D}$ be an arbitrary domain description and W be a world such that $occ_{\mathcal{D}}(W) = \{I_1, \ldots, I_n\} \neq \emptyset$ where $I_1, \ldots, I_n$ are ordered w.r.t. $\leq$, and let c be the c-proposition activated in W at I_n w.r.t. $\mathcal{D}$. Then W is well-behaved w.r.t. $\mathcal{D}$ if and only if (i) there exists a unique world W' well-behaved w.r.t. $\mathcal{D}_{<I_n}$ which is indistinguishable from W up to I_n, (ii) for all $I > I_n$, $W(I){\restriction}\mathcal{F} = \tilde{S}^W_{>I_n}$ where $\tilde{S}^W_{>I_n} = (W(I_n){\restriction}\mathcal{F}) \oplus \chi(O)$ for some outcome $O \in head(c)$, and (iii) W satisfies CWA for actions w.r.t. $\mathcal{D}$.

Furthermore,

$$M_{\mathcal{D}}(W) = \frac{\epsilon_{\mathcal{D}}(W)}{\epsilon_{\mathcal{D}_{<I_n}}(W')} \cdot M_{\mathcal{D}_{<I_n}}(W') \cdot t_{\mathcal{D}}(W(I_n), \tilde{S}^W_{>I_n}) \tag{3.16}$$

Proof. **"Only if" subproof.** Let W be well-behaved w.r.t. $\mathcal{D}$. Let $tr = \langle tr(\boxtimes)@\boxtimes, tr(I_1)@I_1, \ldots, tr(I_n)@I_n\rangle$ be an arbitrary trace of W w.r.t. $\mathcal{D}$ and consider the candidate trace $tr' = \langle tr(\boxtimes)@\boxtimes, tr(I_1)@I_1, \ldots, tr(I_{n-1})@I_{n-1}\rangle$.

Since W is well-behaved w.r.t. $\mathcal{D}$, since $\mathcal{D}$ and $\mathcal{D}_{<I_n}$ differ only by one or more p-propositions occurring at I_n, and since tr' does not mention any instant strictly greater than I_{n-1}, it is possible to construct a world W' which has trace tr' w.r.t. $\mathcal{D}_{<I_n}$ and which is fluent-indistinguishable from W up to instant I_n by simply considering that $W'(\bar{0}){\restriction}\mathcal{F} = \chi(tr'(\boxtimes)) = W(\bar{0}){\restriction}\mathcal{F}$ makes the Initial Condition satisfied w.r.t. $\mathcal{D}_{<I_n}$ as both $\mathcal{D}$ and $\mathcal{D}_{<I_n}$ share the same i-proposition, and a similar argument applies to the Justified Change Condition w.r.t. $\mathcal{D}_{<I_n}$. The Justified Change Condition w.r.t. $\mathcal{D}$ and $\mathcal{D}_{<I_n}$ guarantees that for all instants $I \leq I_n$, $W(I){\restriction}\mathcal{F} = tr(\boxtimes) \oplus \cdots \oplus tr(I_i) = tr'(\boxtimes) \oplus \cdots \oplus tr'(I_i) = W'(I){\restriction}\mathcal{F}$ for some $i < n$, hence W

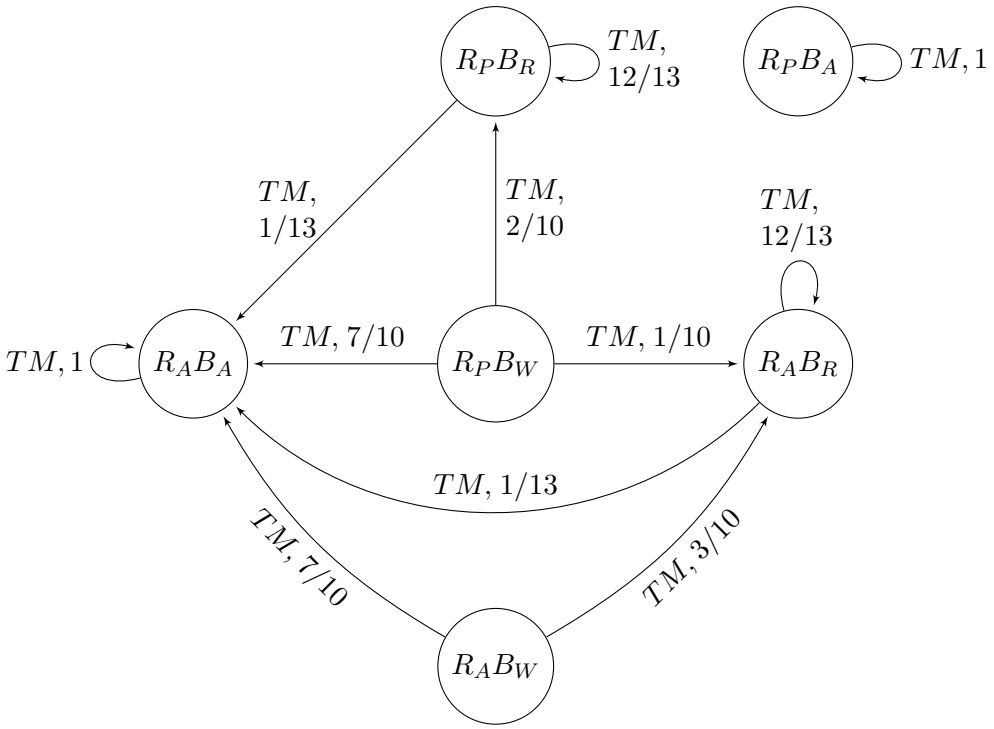


Figure 3.1: Transition function for example 3.7. Node R_PB_R represents the fluent state {Rash=Present, Bacteria=Resistant}, node R_PB_W represents the fluent state {Rash=Present, Bacteria=Weak}, R_AB_A represents the fluent state {Rash=Absent, Bacteria=Absent}, node R_AB_R represents the fluent state {Rash=Absent, Bacteria=Resistant}, node R_PB_A represents the fluent state {Rash=Present, Bacteria=Absent}, and label TM represents {TakesMedicine=$\top$}.

is fluent-indistinguishable from W' up to I_n. Such a W' might not be unique, but choosing W' as to satisfy $A \in W(I) \Leftrightarrow A \in W'(I)$ for all $I < I_n$ and $A \notin W'(I)$ for all $A \in \mathcal{A}$ and $I \geq I_n$ guarantees the uniqueness of W'. Hence, (i) holds. Since W is well-behaved w.r.t. $\mathcal{D}$ and I_n is the greatest element in $occ_{\mathcal{D}}$, $\chi(tr(I_n)) = O$ for some $O \in \mathit{head}(c)$ and Justified Change implies $W(I)\restriction\mathcal{F} = (W(I_n)\restriction\mathcal{F}) \oplus \chi(tr(I_n))$ for all $I > I_n$, and since $\tilde{S}^W_{>I_n}$ is this unique fluent state, (ii) is also satisfied. Finally, (iii) holds since by hypothesis W is well-behaved w.r.t. $\mathcal{D}$.

"If" subproof. Let W' be a well-behaved world w.r.t. $\mathcal{D}_{<I_n}$ and let $occ_{\mathcal{D}_{<I_n}}(W') = \{I_1, \ldots, I_{n-1}\}$. Let $tr' = \langle tr(\boxtimes)@\boxtimes, tr(I_1)@I_1, \ldots, tr(I_{n-1})@I_{n-1}\rangle$ be a trace of W' w.r.t. $\mathcal{D}_{<I_n}$ and construct the candidate trace $tr = \langle tr(\boxtimes)@\boxtimes, tr(I_1)@I_1, \ldots, O@I_n\rangle$ for the outcome $O \in \mathit{head}(c)$ such that $(W(I)\restriction\mathcal{F}) = (W(I_n)\restriction\mathcal{F}) \oplus \chi(O)$ for all $I > I_n$. Since W' is well-behaved w.r.t. $\mathcal{D}_{<I_n}$ and indistinguishable from W up to I_n by hypothesis (i), it is possible to conclude that tr is a trace of W' w.r.t. $\mathcal{D}$ by noticing again that both $\mathcal{D}$ and $\mathcal{D}_{<I_n}$ share the same i-proposition, and a similar argument applies for the Justified Change Condition w.r.t. $\mathcal{D}$ (also using hypothesis (ii)). Since W also satisfies CWA for actions by hypothesis (iii), it is well-behaved.

"Furthermore" subproof. Let $\tilde{S}^W_{>I_n}$ and c be as in the statement of the proposition. The above proof implies that any trace tr of W w.r.t. $\mathcal{D}$ can be constructed from a trace tr' of an appropriate W' by letting $tr(I_i) = tr'(I_i)$ for $i < n$ and $tr(I_n) = \chi(O)$ for some $O \in \mathit{head}(c)$ such that $\tilde{S}^W_{>I_n} = (W(I_n)\restriction\mathcal{F}) \oplus \chi(O)$ (and notice that there is at least one such outcome O

since W is well-behaved), i.e. for some $O \in \mathit{tset}_{\mathcal{D}}(W(I_n), \tilde{S}^W_{>I_n})$.

Definition 3.32 now implies

$$\begin{aligned} M_{\mathcal{D}}(W) &= \epsilon_{\mathcal{D}}(W) \cdot \sum_{tr \in \mathcal{TR}^W_{\mathcal{D}}} \epsilon(tr) \\ &= \epsilon_{\mathcal{D}_{<I_n}}(W') \cdot \frac{\epsilon_{\mathcal{D}}(W)}{\epsilon_{\mathcal{D}_{<I_n}}(W')} \cdot \pi(\mathit{tset}_{\mathcal{D}}(W(I_n), \tilde{S}^W_{>I_n})) \cdot \sum_{tr' \in \mathcal{TR}^{W'}_{\mathcal{D}_{<I_n}}} \epsilon(tr') \\ &= \frac{\epsilon_{\mathcal{D}}(W)}{\epsilon_{\mathcal{D}_{<I_n}}(W')} \cdot t_{\mathcal{D}}(W(I_n), \tilde{S}^W_{>I_n}) \cdot \left(\epsilon_{\mathcal{D}_{<I_n}}(W') \cdot \sum_{tr' \in \mathcal{TR}^{W'}_{\mathcal{D}_{<I_n}}} \epsilon(tr') \right) \\ &= \frac{\epsilon_{\mathcal{D}}(W)}{\epsilon_{\mathcal{D}_{<I_n}}(W')} \cdot t_{\mathcal{D}}(W(I_n), \tilde{S}^W_{>I_n}) \cdot M_{\mathcal{D}_{<I_n}}(W'). \end{aligned}$$

which is well defined since $\epsilon(N, W) > 0$ for any action narrative N and world W. □

Corollary 3.2.1. Let $\mathcal{D}$ be any domain description and let I be any instant. Then W is well-behaved w.r.t. $\mathcal{D}_{\leq I}$ if and only if (i) there exists a unique world W' well-behaved w.r.t. $\mathcal{D}_{<I}$ which is indistinguishable from W up to I, (ii) for all $I' > I$ then $W(I')\restriction\mathcal{F} = \tilde{S}^W_{>I}$ where

$$\tilde{S}^W_{>I} = \begin{cases} (W(I)\restriction\mathcal{F}) \oplus \chi(O) \text{ for some } O \in \mathit{head}(\mathit{cprop}_{\mathcal{D}_{\leq I}}(W, I)) & \text{if } I \in \mathit{occ}_{\mathcal{D}_{\leq I}}(W) \\ W(I)\restriction\mathcal{F} & \text{otherwise} \end{cases}$$

and (iii) W satisfies CWA for actions w.r.t. $\mathcal{D}_{\leq I}$.

Furthermore, for $\tilde{S}^W_{>I_n}$ the unique fluent state taken by W at instants $I > I_n$:

$$M_{\mathcal{D}_{\leq I}}(W) = \frac{\epsilon_{\mathcal{D}_{\leq I}}(W)}{\epsilon_{\mathcal{D}_{<I}}(W')} \cdot M_{\mathcal{D}_{<I}}(W') \cdot t_{\mathcal{D}}(W(I), \tilde{S}^W_{>I}) \tag{3.17}$$

Proof. If $I \in \mathit{occ}_{\mathcal{D}_{\leq I}}(W)$ then the corollary follows directly from proposition 3.2 since the domain description $\mathcal{D}_{\leq I}$ satisfies all of its hypotheses, so only need to consider the case in which $I \notin \mathit{occ}_{\mathcal{D}_{\leq I}}(W)$. Proof (of the "if", "only if" and "furthermore" parts) is similar to that of proposition 3.2, in the easier case where the two worlds W and W' can be taken to have the exact same trace as no c-proposition is activated in W at I.

□

Lemma 3.1 (Transition Function Normalisation)**.** For any $\mathcal{D}$ and any state S,

$$\sum_{\tilde{S}' \in \tilde{\mathcal{S}}} t_{\mathcal{D}}(S, \tilde{S}') = 1.$$

Proof. By cases:

Case 1. If there is no c-proposition c such that S entails $\mathit{body}(c)$, then it follows from

definition 3.38 that

$$\sum_{\tilde{S}' \in \tilde{\mathcal{S}}} t_{\mathcal{D}}(S, \tilde{S}') = t_{\mathcal{D}}(S, S{\restriction}\mathcal{F}) = \pi((\emptyset, 1)) = 1$$

Case 2. Let c be the unique c-proposition S entails $\mathit{body}(c)$. Then, applying the definition of $t_{\mathcal{D}}$ from definition 3.38 gives

$$\sum_{\tilde{S}' \in \tilde{\mathcal{S}}} t_{\mathcal{D}}(S, \tilde{S}') = \sum_{\tilde{S}' \in \tilde{\mathcal{S}}} \pi(\mathit{tset}_{\mathcal{D}}(S, \tilde{S}')) \tag{3.18}$$

Notice that for a fixed outcome O, it is impossible to have $O \in \mathit{tset}_{\mathcal{D}}(S, \tilde{S}')$ and $O \in \mathit{tset}_{\mathcal{D}}(S, \tilde{S}'')$ for two distinct fluent states $\tilde{S}', \tilde{S}''$ as this would imply $\tilde{S}' = (S{\restriction}\mathcal{F}) \oplus \chi(O) = \tilde{S}''$. Hence it is sufficient to show that $\{O \in \mathit{tset}_{\mathcal{D}}(S, \tilde{S}') \mid \tilde{S}' \in \tilde{\mathcal{S}}\} = \mathit{head}(c)$, as this implies that the sum (3.18) equals 1 since $\pi(\mathit{head}(c)) = 1$ by definition of a c-proposition.

By definition of a transition set, $\{O \in \mathit{tset}_{\mathcal{D}}(S, \tilde{S}') \mid \tilde{S}' \in \tilde{\mathcal{S}}\} \subseteq \mathit{head}(c)$. Conversely, for any $O \in \mathit{head}(c)$, $O \in \mathit{tset}_{\mathcal{D}}(S, \tilde{S}')$ for $\tilde{S}' = (S{\restriction}\mathcal{F}) \oplus \chi(O)$, hence $\mathit{head}(c) \subseteq \{O \in \mathit{tset}_{\mathcal{D}}(S, \tilde{S}') \mid \tilde{S}' \in \tilde{\mathcal{S}}\}$ which ends the proof of lemma. □

Lemma 3.2 (Action Narrative Normalisation)**.** Let $\mathcal{D}$ be any domain description, I be any instant and N_I be the (possibly empty) action narrative that contains exactly those p-propositions in $\mathcal{D}$ that have instant I. Let $\{W_1, \ldots, W_m\}$ be a maximal set of well-behaved worlds w.r.t. $\mathcal{D}$ such that $W_i(I){\restriction}\mathcal{F} = W_j(I){\restriction}\mathcal{F}$ for all $1 \leq i, j \leq m$ and $W_i(I) \neq W_j(I)$ when $i \neq j$. Then,

$$\sum_{j=1}^{m} \epsilon(N_I, W_j) = 1$$

Proof. Fix a maximal set $\{W_1, \ldots, W_m\}$ as in the hypothesis and let $\tilde{S}$ be the fluent state such that $\tilde{S} = W_i(I){\restriction}\mathcal{F}$ for all $1 \leq i \leq m$. Let $\{p_1, \ldots, p_k\}$ be a maximal set of p-propositions in N_I such that $\tilde{S} \models \theta_i$ and $P_i^+ < 1$ for all $1 \leq i \leq k$, where each p_i has the form

$$A_i \textbf{ performed-at } I \textbf{ with-prob } P_i^+ \textbf{ if-holds } \theta_i$$

From maximality of $\{W_1, \ldots, W_m\}$ and CWA for actions it follows that $m = 2^k$, with every world in this set having a different assignment of actions $A_1, \ldots, A_k$ to truth values at instant I. Let $\bar{x} = \langle x_1, \ldots, x_k \rangle$ be a k-dimensional vector representing a specific assignment of $A_1, \ldots, A_k$ to truth values, where each $x_i \in \{0, 1\}$. Therefore the sum $\sum_{j=1}^{m} \epsilon(N_I, W_j)$ evaluates to:

$$\sum_{\bar{x}} \left(\prod_{i=1}^{k} (P_i^+)^{x_i} \left(1 - P_i^+\right)^{1 - x_i} \right) = \prod_{i=1}^{k} \left(P_i^+ + (1 - P_i^+) \right) = 1$$

□

Definition 3.39 (hasInstant)**.** For a p-proposition p, $\mathit{hasInstant}(p)$ stands for the instant p has,

and for an action narrative N this is extended to:

$$\mathit{hasInstants}(N) = \bigcup_{p \in N} \mathit{hasInstant}(p)$$

At this point, all the machinery to prove some important properties of the model of a domain description is available.

The following proposition shows that adding p-propositions having instants $I \geq I'$ (for some fixed $I0$) to a a domain description $\mathcal{D}$ does not affect the value of the model of i-formulas $[\theta]@I''$ for $I'' \leq I'$.

Proposition 3.3 (Causality)**.** Let $\mathcal{D}$ and $\mathcal{D}' = \mathcal{D} \cup N_{I'}$ be two domain descriptions, where $N_{I'}$ is an action narrative such that all p-propositions in $N_{I'}$ have instant $I' > \max_{I \in \mathcal{I}} \mathit{hasInstants}(\mathit{narr}(\mathcal{D}))$ if $\mathit{narr}(\mathcal{D})$ is non-empty, and it can take any value otherwise. Then, for any $I < I'$ and any formula θ,

$$M_{\mathcal{D}'}([\theta]@I) = M_{\mathcal{D}}([\theta]@I)$$

Proof. For a generic domain description $\mathcal{D}$, let $[W']^I_{\mathcal{D}}$ be the set of well-behaved worlds w.r.t. $\mathcal{D}$ that are indistinguishable from W' up to I. Corollary 3.2.1 and lemma 3.1 yield:

$$\begin{aligned}
M_{\mathcal{D}'}([\theta]@I) &= \sum_{W \models [\theta]@I} M_{\mathcal{D}'}(W) \\
&\overset{\text{Cor.3.2.1}}{=} \sum_{W' \models [\theta]@I} \sum_{W \in [W']^{I'}_{\mathcal{D}'}} \frac{\epsilon_{\mathcal{D}'}(W)}{\epsilon_{\mathcal{D}}(W')} \cdot M_{\mathcal{D}}(W') \cdot t_{\mathcal{D}'}(W(I'), \tilde{S}^W_{>I'}) \\
&= \sum_{W' \models [\theta]@I} M_{\mathcal{D}}(W') \sum_{W \in [W']^{I'}_{\mathcal{D}'}} \frac{\epsilon_{\mathcal{D}'}(W)}{\epsilon_{\mathcal{D}}(W')} \cdot t_{\mathcal{D}'}(W(I'), \tilde{S}^W_{>I'})
\end{aligned}$$

According to corollary 3.2.1 every world in $[W']^{I'}_{\mathcal{D}'}$ can be reconstructed from its state at instant I' and the unique state $\tilde{S}'$ that it takes at instants strictly greater than I'. Let $\{W_1, \ldots, W_m\}$ be a maximal set of well-behaved worlds such that $W_i(I) = W_j(I)$ when $i \neq j$ for $1 \leq i, j \leq m$. Then, the above chain of equalities continues as follows:

$$\begin{aligned}
&= \sum_{W' \models [\theta]@I} M_{\mathcal{D}}(W') \sum_{j=1}^{m} \frac{\epsilon_{\mathcal{D}'}(W_j)}{\epsilon_{\mathcal{D}}(W')} \cdot \sum_{\tilde{S}' \in \tilde{\mathcal{S}}} t_{\mathcal{D}'}(W_j(I'), \tilde{S}') \\
&\overset{\text{Lem.3.1}}{=} \sum_{W' \models [\theta]@I} M_{\mathcal{D}}(W') \sum_{j=1}^{m} \frac{\epsilon_{\mathcal{D}'}(W_j)}{\epsilon_{\mathcal{D}}(W')} \\
&\overset{\text{Lem.3.2}}{=} \sum_{W' \models [\theta]@I} M_{\mathcal{D}}(W') = M_{\mathcal{D}}([\theta]@I)
\end{aligned}$$

and notice that it is possible to apply lemma 3.2 since $\frac{\epsilon_{\mathcal{D}'}(W_j)}{\epsilon_{\mathcal{D}}(W')} = \epsilon(N_{I'}, W_j)$. □

Proposition 3.4 (Model is a Probability Distribution)**.** Let $\mathcal{D}$ be any domain description. Then,

$\models \mathcal{D}$ is a probability distribution in the sense that it satisfies eq. (2.3) from section 2.2.1, i.e.

$$\sum_{W \in \mathcal{W}} M_{\mathcal{D}}(W) = 1$$

Proof. Need to show that

$$\sum_{W \in \mathcal{W}} M_{\mathcal{D}}(W) = \sum_{W \in \mathcal{W}_{\mathcal{D}}} M_{\mathcal{D}}(W) \tag{3.19}$$

evaluates to 1, where the second equality is guaranteed by the fact that $M_{\mathcal{D}}(W) = 0$ when W is not well-behaved.

(3.19) is proved by induction on the size of *hasInstants*(*narr*($\mathcal{D}$)) $= \{I_1, \ldots, I_n\}$ where $I_1, \ldots, I_n$ are ordered w.r.t. $\leq$.

Base case. Consider $\mathcal{D}_\emptyset$ first. Since there are no p-propositions in $\mathcal{D}_\emptyset$, *hasInstants*($\mathcal{D}_\emptyset$) $= \emptyset$, $\epsilon_{\mathcal{D}_\emptyset}(W) = \emptyset$ for all worlds W, and the sum (3.19) becomes:

$$\sum_{W \in \mathcal{W}_{\mathcal{D}_\emptyset}} \left(\sum_{tr \in \mathcal{TR}^W_{\mathcal{D}_\emptyset}} \pi(tr(\boxtimes)) \right) \tag{3.20}$$

Let "**initially-one-of** $\{O_1, \ldots, O_m\}$" be the (unique) i-proposition in $\mathcal{D}_\emptyset$. As a first step, it is shown the well-behaved worlds w.r.t. $\mathcal{D}_\emptyset$ are exactly those Ws taking the form $W(I)\restriction\mathcal{F} = \chi(O_i)$ and $\neg A \in W(I)$ for all instants I and all action symbols A.

If W has this form, then it satisfies CWA for actions (as there are no p-propositions in $\mathcal{D}_\emptyset$, and this is consistent with $\neg A \in W(I)$ for all I and A), it satisfies the Initial Condition w.r.t. $\mathcal{D}_\emptyset$ as O_i is an initial choice w.r.t. $\mathcal{D}_\emptyset$ and $W(\bar{0})\restriction\mathcal{F} = \chi(O_i)$ by definition, and finally it also satisfies the Justified Change Condition in the form (3.7) as $occ_{\mathcal{D}_\emptyset}(W) = \emptyset$, which in turn forces $W(I)\restriction\mathcal{F} = W(I')\restriction\mathcal{F}$ for all I and I'. The proof that if W is well-behaved then it is of the form above is an inversion of the previous chain of implications.

Notice that each of these well-behaved worlds is consistent with a unique trace $\langle O_i @ \boxtimes \rangle$ for some $1 \leq i \leq m$, and let W_i denote the world having trace $\langle O_i @ \boxtimes \rangle$ for $1 \leq i \leq m$. Hence $\mathcal{W}_{\mathcal{D}_\emptyset} = \{W_1, \ldots, W_m\}$. For such W_i,

$$\sum_{tr \in \mathcal{TR}^{W_i}_{\mathcal{D}_\emptyset}} \pi(tr(\boxtimes)) = \pi(O_i)$$

and (3.20) evaluates to:

$$\sum_{W_i \in \mathcal{W}_{\mathcal{D}_\emptyset}} \pi(O_i) = \sum_{i=1}^{m} \pi(O_i) = 1 \tag{3.21}$$

as $\pi(\{O_1, \ldots, O_m\}) = 1$ by definition 3.9 of an i-proposition.

Inductive step. Assume that $M_{\mathcal{D}_{<I_i}}(\top) = 1$ for some $i \leq n$. Using proposition 3.3, $M_{\mathcal{D}_{\leq I_i}}(\top) = M_{\mathcal{D}_{<I_i}}(\top) = 1$.

□

An immediate consequence of the previous proposition is the following one:

Corollary 3.4.1. For any given domain description $\mathcal{D}$, $\mathcal{W}_\mathcal{D} \neq \emptyset$.

Notice that since $\sum_{W|\models\top} M_\mathcal{D}(W) = 1$ for any domain description $\mathcal{D}$, it follows that for any i-formula φ and any tautological i-formula $\top$ the following holds:

$$M_\mathcal{D}(\varphi) = M_\mathcal{D}(\varphi \mid \top)$$

Finally, the following is the central result about $M_\mathcal{D}$:

Proposition 3.5. Given a model $M_\mathcal{D}$ of a domain description $\mathcal{D}$, its extension to $M_\mathcal{D}$ is a probability function.

Proof. It is sufficient to show that for any domain description, requirements 1 and 2 as in definition 3.35 are always satisfied by a model of that domain description. Requirement 1 follows directly from proposition 3.4. For requirement 2, let φ and ψ be two i-formulas such that $\varphi \mid\!\models \neg\psi$. Obviously, since $\varphi \mid\!\models \neg\psi$ if for some $W \in \mathcal{W}$, $W \mid\!\models \varphi$, then $W \mid\!\not\models \psi$ and vice-versa, hence

$$M^*_\mathcal{D}(\varphi \vee \psi) = \sum_{W|\models\varphi\vee\psi} M_\mathcal{D}(W) = \sum_{W|\models\varphi} M_\mathcal{D}(W) + \sum_{W|\models\psi} M_\mathcal{D}(W) = M^*_\mathcal{D}(\varphi) + M^*_\mathcal{D}(\psi).$$

□

Corollary 3.5.1. If ψ is an i-formula such that $M_\mathcal{D}(\psi) \neq 0$, then $M_\mathcal{D}(\cdot \mid \psi)$ is a probability function.

Proof. Follows directly from property 6 from proposition 2.1 and proposition 3.5. □

3.4 Example entailments

This section provides some example entailments from domain descriptions introduced in this chapter.

Example 3.22. $\mathcal{D}_C$ as in example 3.6 entails, among others:

(|⊨C1) $\top$ **holds-with-prob** 1

(|⊨C2) $[\mathit{Coin}=\mathit{Tails}]@0$ **holds-with-prob** 0

(|⊨C3) $[\mathit{Toss}=\top]@1$ **holds-with-prob** 1

(|⊨C4) $[\mathit{Coin}=\mathit{Heads}]@2$ **holds-with-prob** 0.51

(|⊨C5) $[\mathit{Coin}=\mathit{Heads}]@1 \wedge [\mathit{Coin}=\mathit{Tails}]@3$
holds-with-prob 0.49

Example 3.23 (Decay and Persistence)**.** Consider the following (abstract) domain description $\mathcal{D}_{DP}$:

(DP1) F **takes-values** $\{\top, \bot\}$

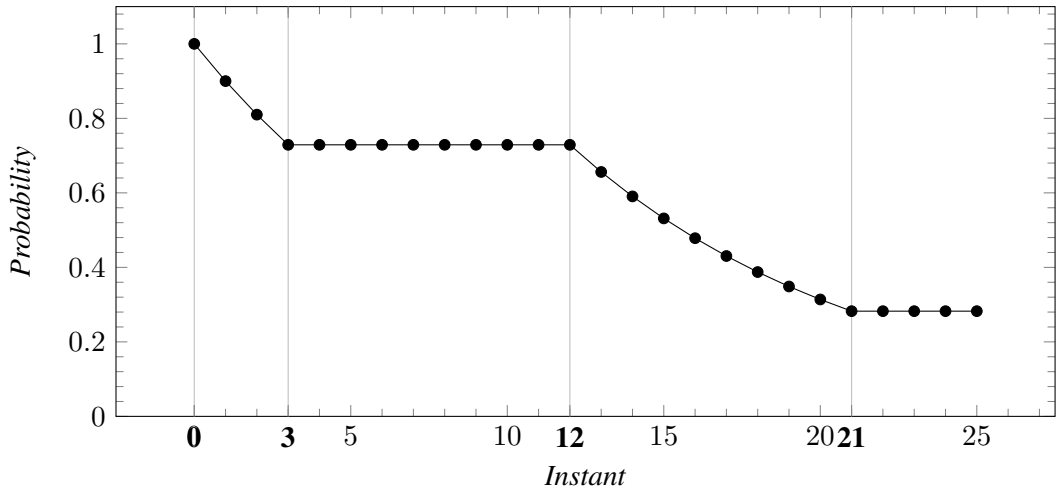


Figure 3.2: Probability of $[F=\top]@I$ *in* $\mathcal{D}_{DP}$ *as in example 3.23*

(DP2) **initially-one-of** $\{(\{F=\top\},1)\}$

(DP3) A **causes-one-of** $\{(\{F=\bot\},1/10)\}$

(DP4) A **performed-at** I, for $I=\{0,1,2\}$ and $I=\{12,\ldots,20\}$

$\mathcal{D}_{DP}$ entails the h-proposition "$[F=\top]@I$ **holds-with-prob** P" for the following instant-probability pairs (see fig. 3.2): $(0,1)$, $(1,0.9)$, $(2,0.81)$, $(3,0.729)$, $(4,0.729)$, $(5,0.729)$, $(6,0.729)$, $(7,0.729)$, $(8,0.729)$, $(9,0.729)$, $(10,0.729)$, $(11,0.729)$, $(12,0.729)$, $(13,0.6561)$, $(14,0.59049)$, $(15,0.531441)$, $(16,0.478297)$, $(17,0.430467)$, $(18,0.38742)$, $(19,0.348678)$, $(20,0.313811)$ and $(I,0.28243)$, for all $I>20$.

This provides a first glimpse at how decay and persistence work in PEC. As it is evident from the figure, repeated occurrence of action A from instant 0 to instant 2 and then again from instant 12 to instant 20 makes the probability of $F=\top$ decay in intervals $[0,3]$ and $[12,21]$. A similar mechanism is exploited in the Tuberculosis example to make the probability of *Tuberculosis*$=$*Latent* decay. When no action is performed, the probability of $F=\top$ does not change as an effect of persistence, see intervals $[3,12]$ and $[21,+\infty]$.

Example 3.24. The following h-propositions are entailed by $\mathcal{D}_A$ as in example 3.7:

($|\!=$A1) $[$*Bacteria*$=$*Weak*$]@0$ **holds-with-prob** 0.9

($|\!=$A2) $[$*Bacteria*$=$*Weak* $\wedge$ *Rash*$=$*Absent*$]@0$
holds-with-prob 0

($|\!=$A3) $[$*Bacteria*$=$*Resistant*$]@2$ **holds-with-prob** 0.27

($|\!=$A4) $[$*Rash*$=$*Absent*$]@4$ **holds-with-prob** 0.733846

($|\!=$A5) $[$*Bacteria*$=$*Absent* $\wedge$ *Rash*$=$*Absent*$]@4$
holds-with-prob 0.650769

Notice that from $||\!\models$*A4* and $||\!\models$*A5* it is possible to calculate the conditional probability that the medicine has cured the infection at instant 4, i.e. $[$*Bacteria*$=$*Absent*$]@4$, given that no sign of rash is visible at the end of the treatment, i.e. $[$*Rash*$=$*Absent*$]@4$. Applying eq. (3.15) gives that this probability equals $0.650769/0.733846 = 0.8867923243$. Therefore, using the syntax for conditional entailment defined in definition 3.34,

$$\mathcal{D} \,||\!\models \textbf{given}\ [\textit{Rash} = \textit{Absent}]@4, [\textit{Bacteria} = \textit{Absent}]@4\ \textbf{holds-with-prob}\ 0.8867923243$$

or equivalently

$$(\mathcal{D} \mid [\textit{Rash} = \textit{Absent}]@4) \,||\!\models [\textit{Bacteria} = \textit{Absent}]@4\ \textbf{holds-with-prob}\ 0.8867923243$$

3.5 Summary

This chapter introduces PEC+, an Event Calculus style action language for reasoning about probabilistic causal and narrative information. Its action language style syntax, defined in section 3.1, is similar to that of Language $\mathcal{E}$ [36] and $\mathcal{M}$odular-$\mathcal{E}$ [37]. Its semantics is given in terms of possible worlds which constitute possible evolutions of the domain starting from an initial state, and builds on that of Epistemic Functional Event Calculus [42]. These worlds are then assigned a weight by means of a *model* function (see definition 3.32) which represents the degree of plausibility these worlds have. This model function is then shown to satisfy some properties, most importantly that it is a probability distribution (see proposition 3.4).

Chapter 4

ASP Implementation of PEC+

This chapter describes an implementation of PEC+. It presents a translation procedure from PEC+ domain descriptions into ASP programs, and proves its correctness under stable models semantics.

The idea is of first transforming a domain description into an ASP program in a such a way that there is a one-to-one correspondence between answer sets of the ASP program and traces of the domain description. This transformation is described in sections 4.1 and 4.2 and its correctness is demonstrated in section 4.3. In order to compute entailments of h-propositions, these ASP programs are then augmented with an ASP translation of the i-formula contained in the h-proposition of interest, called a *query*, and the answer sets of these augmented ASP programs correspond to those traces of the domain description compatible with the query. Some examples and brief remarks about the efficiency of this computational method conclude the chapter in section 4.4.

For the purposes of translation, it is assumed that the underlying PEC+ domain language has been adjusted so that the set $\mathcal{I}$ of instants is a finite interval $\{0, 1, \ldots, \mathit{maxinst}\}$ of $\mathbb{N}$, with $\bar{0} = 0$ and $\leq = \leq_{\mathbb{N}}$ being the usual ordering relation between naturals. Furthermore, since in logic programming lowercase letters are conventionally used for constants, we switch to that convention by letting lower case letters be the logic programming counterparts of (upper case) constants in PEC+ so that e.g. *coin* is the translation of the fluent *Coin* and f is the translation of fluent F. Literals of the form $X = V$ are translated into pairs of the form (x, v).

4.1 Domain-dependent part of the translation

4.1.1 An example translation

To aid the reader's intuition, in this subsection we list the full domain-dependent part of the translation of $\mathcal{D}_C$ as in example 3.6.

Example 4.1 (Translation of the Coin Toss Domain)**.** Let $\mathcal{D}_C$ be as in example 3.6. The translation of $\mathcal{D}_C$ results in the following set of axioms:

(TC0) $\mathit{fluent}(\mathit{coin})$.
$\mathit{action}(\mathit{toss})$.
$\mathit{instant}(0..\mathit{maxinst})$.

(TC1) $\mathit{possVal}(\mathit{coin}, \mathit{heads})$.
$\mathit{possVal}(\mathit{coin}, \mathit{tails})$.

(TC2) $\mathit{belongsTo}((\mathit{coin}, \mathit{heads}), id_1^0)$.
$\mathit{initialCondition}((id_1^0, 1))$.

(TC3) $\mathit{belongsTo}((\mathit{coin}, \mathit{heads}), id_1^1)$.
$\mathit{causedOutcome}((id_1^1, 49/100), I) \leftarrow$
$\mathit{holds}(((\mathit{toss}, \mathit{true}), I))$.

(TC4) $\mathit{belongsTo}((\mathit{coin}, \mathit{tails}), id_2^1)$.
$\mathit{causedOutcome}((id_2^1, 49/100), I) \leftarrow$
$\mathit{holds}(((\mathit{toss}, \mathit{true}), I))$.

(TC5) $\mathit{causedOutcome}((id_3^1, 2/100), I) \leftarrow$
$\mathit{holds}(((\mathit{toss}, \mathit{true}), I))$.

(TC6) $\mathit{performed}(\mathit{toss}, 1, 1)$.

where, informally, the set of axioms *TC0* is the translation of the three sorts $\mathcal{F}$, $\mathcal{A}$ and $\mathcal{I}$; axioms *TC1*, *TC2* and *TC6* are the translation of propositions *C1*, *C2* and *C4* respectively; axioms *TC3* to *TC5* together give the translation of the c-proposition *C3*, with each of them corresponding to one of its outcomes. The terms $id_1^0, id_1^1, id_2^1, id_3^1$ are new terms in the program as explained in the next section.

4.1.2 The general domain-dependent translation procedure

The three sorts $\mathcal{F}$, $\mathcal{A}$ and $\mathcal{I}$ are translated to the three sets $\{\mathit{fluent}(f) \mid F \in \mathcal{F}\}$, $\{\mathit{action}(a) \mid A \in \mathcal{A}\}$ and $\{\mathit{instant}(i) \mid I \in \mathcal{I}\}$ respectively (see e.g. axiom *TC0* in example 4.1).

Let c be a c-proposition of the form (3.2):

$$\theta \textbf{ causes-one-of } \{O_1, O_2, \ldots, O_m\}$$

but first considering the case where θ is a conjunction of the form $X_1 = V_1 \wedge \cdots \wedge X_j = V_j$. Given a conjunction θ of this form, we write $\mathit{holds}([\theta]@I)$ as a shorthand for the logic programming conjunction

$$\mathit{holds}(((x_1, v_1), i)), \ldots, \mathit{holds}(((x_j, v_j), i)).$$

We fix an enumeration (without repetitions) of all the c-propositions in $\mathcal{D}$. Let the c-proposition c referred to above be the nth proposition occurring in this enumeration. Then, c is translated to:

$$\{\, \mathit{belongsTo}((x, v), id_i^n) \mid i = 1, \ldots, m, X = V \in \chi(O_i) \,\} \cup$$
$$\{\, \mathit{causedOutcome}((id_i^n, p), I) \leftarrow \mathit{holds}([\theta]@I) \mid i = 1, \ldots, m,\ P = \pi(O_i) \,\}$$

where $id_1^n, \ldots, id_m^n$ are new constants in the underlying ASP language. Each of these constants id_i^n represents the partial fluent state $\chi(O_i)$ in the body of the nth c-proposition, and *belongsTo* represents the $\in$ relation between literals and partial fluent states.

Example 4.2. Axioms *TC3* to *TC5* are the translation of the c-proposition *C3* from example 3.6.

As a further example, consider the c-proposition *A5* in example 3.7, and notice that two outcomes occur in it, i.e. $(\{\mathit{Bacteria}{=}\mathit{Absent}, \mathit{Rash} = \mathit{Absent}\}, 1/13)$ and $(\emptyset, 4/13)$. If we fix the enumeration of c-propositions in $\mathcal{D}_A$ such that proposition *A4* is first and proposition *A5* is second, *A5* is translated to:

(TA1) $\mathit{belongsTo}((\mathit{bacteria}, \mathit{absent}), id_1^2).$
$\mathit{belongsTo}((\mathit{rash}, \mathit{absent}), id_1^2).$
$\mathit{causedOutcome}((id_1^2, 1/13), I) \leftarrow$
$\mathit{holds}((\mathit{takesMedicine}, \mathit{true}), I),$
$\mathit{holds}((\mathit{bacteria}, \mathit{resistant}), I).$

(TA2) $\mathit{causedOutcome}((id_2^2, 4/13), I) \leftarrow$
$\mathit{holds}((\mathit{takesMedicine}, \mathit{true}), I),$
$\mathit{holds}((\mathit{bacteria}, \mathit{resistant}), I).$

If θ is not a conjunction of literals, then we represent it in Disjunctive Normal Form, i.e. in the form $\theta_1 \vee \cdots \vee \theta_r$ where $\theta_1, \ldots, \theta_r$ are conjunctions of literals, and then translate c to

$$\{\, \mathit{belongsTo}((x, v), id_i^n) \mid i = 1, \ldots, m, X = V \in \chi(O_i) \,\} \cup$$
$$\{\, \mathit{causedOutcome}((id_i^n, p), I) \leftarrow \mathit{holds}([\theta_1]@I);\ \ldots;\ \mathit{holds}([\theta_r]@I) \mid i = 1, \ldots, m,\ P = \pi(O_i) \,\}$$

We write $C_{\mathcal{D}}$ for the set of all translated c-propositions in $\mathcal{D}$.

The translation of i-propositions works in a very similar way: if J is an i-proposition of the general form (3.3):

$$\textbf{initially-one-of}\ \{O_1, O_2, \ldots, O_m\}$$

then its translation is given by the following set of axioms:

$$\{\, \mathit{belongsTo}((x, v), id_i^0) \mid i = 1, \ldots, m, X = V \in \chi(O_i) \,\} \cup$$
$$\{\, \mathit{initialCondition}((id_i^0, p)) \mid i = 1, \ldots, m, P = \pi(O_i) \,\}$$

and we write $I_{\mathcal{D}}$ for the set of all translated i-propositions in $\mathcal{D}$.

Example 4.3. An example of a translated i-proposition is the set of axioms *TC2*, that translate the i-proposition *C2* as in example 4.1.

The i-proposition *A3* from example 3.7 is translated to:

(TA3) $\mathit{belongsTo}((\mathit{bacteria}, \mathit{weak}), id_1^0).$
$\mathit{belongsTo}((\mathit{rash}, \mathit{present}), id_1^0).$
$\mathit{initialCondition}((id_1^0, 9/10)).$

(TA4) $\mathit{belongsTo}((\mathit{bacteria}, \mathit{absent}), id_2^0)$.
$\mathit{belongsTo}((\mathit{rash}, \mathit{present}), id_2^0)$.
$\mathit{initialCondition}((id_2^0, 1/10))$.

Any p-proposition of the form (3.4) is translated to:

$$\mathit{performed}(a, i, p^+) \leftarrow \mathit{holds}([\theta]@I).$$

and we write $P_{\mathcal{D}}$ for the set of all translated p-propositions in $\mathcal{D}$,

Finally, any v-proposition of the form (3.1) is translated to:

$$\{\, \mathit{possVal}(f, v_i) \mid 1 \leq i \leq n \,\}$$

and we write $V_{\mathcal{D}}$ for the set of all translated v-propositions in $\mathcal{D}$.

Example 4.4. The v-proposition *C1* and p-proposition *C4* from from example 3.6 are translated to axioms *TC1* and *TC6* in example 4.1 respectively, while propositions *A1*, *A2*, *A6*, *A7* from example 3.7 are translated to:

(TA5) $\mathit{possVal}(\mathit{bacteria}, \mathit{weak})$.
$\mathit{possVal}(\mathit{bacteria}, \mathit{resistant})$.
$\mathit{possVal}(\mathit{bacteria}, \mathit{absent})$.

(TA6) $\mathit{possVal}(\mathit{rash}, \mathit{present})$.
$\mathit{possVal}(\mathit{rash}, \mathit{absent})$.

(TA7) $\mathit{performed}(\mathit{takesMedicine}, 1, 1)$.

(TA8) $\mathit{performed}(\mathit{takesMedicine}, 3, 1)$.

Finally, p-propositions from the schema *T5* are translated to:

(TT1) $\mathit{performed}(\mathit{reactivation}, i, 8/10^4) \leftarrow$
$\mathit{holds}(((\mathit{tuberculosis}, \mathit{latent}), i))$.

for $i \in \{0, \ldots, 50\}$.

We write $\Pi_{\mathcal{D}}$ for the set of translated propositions from $\mathcal{D}$, e.g. if $\mathcal{D}_C$ is as in example 3.6, $\Pi_{\mathcal{D}_C} = \{\mathit{TC0}, \mathit{TC1}, \ldots, \mathit{TC6}\}$.

4.2 Domain-independent part of the translation

We define the domain-independent part of our theory to be:

(PEC1) $\mathit{possVal}(A, \mathit{true}) \leftarrow \mathit{action}(A)$.
$\mathit{possVal}(A, \mathit{false}) \leftarrow \mathit{action}(A)$.

(PEC2) $\mathit{fluentOrAction}(X) \leftarrow \mathit{fluent}(X)$; $\mathit{action}(X)$.

(PEC3) $\mathit{literal}((X,V)) \leftarrow \mathit{possVal}(X,V).$

(PEC4) $\mathit{iLiteral}(((X,V),I)) \leftarrow \mathit{possVal}(X,V),\ \mathit{instant}(I).$

(PEC5) $\mathit{definitelyPerformed}(A,I) \leftarrow \mathit{performed}(A,I,1).$

(PEC6) $\mathit{possiblyPerformed}(A,I) \leftarrow \mathit{performed}(A,I,P).$

(PEC7) $1\{\ \mathit{holds}(((X,V),I)) : \mathit{iLiteral}(((X,V),I))\ \}1$
$\leftarrow \mathit{instant}(I),\ \mathit{fluentOrAction}(X).$

(PEC8) $\mathit{inOcc}(I) \leftarrow \mathit{instant}(I),\ \mathit{causedOutcome}(O,I).$

(PEC9) $1\{\ \mathit{effectChoice}(O,I) : \mathit{causedOutcome}(O,I)\ \}1$
$\leftarrow \mathit{inOcc}(I).$

(PEC10) $1\{\ \mathit{initialChoice}(O) : \mathit{initialCondition}(O)\ \}1.$

(PEC11) $\bot \leftarrow \mathit{action}(A),\ \mathit{instant}(I),$
$\mathit{holds}(((A,\mathit{true}),I)),\ \mathit{not\ possiblyPerformed}(A,I).$

(PEC12) $\bot \leftarrow \mathit{action}(A),\ \mathit{instant}(I),$
$\mathit{holds}(((A,\mathit{false}),I)),\ \mathit{definitelyPerformed}(A,I).$

(PEC13) $\bot \leftarrow \mathit{initialChoice}((S,P)),\ \mathit{literal}(L),$
$\mathit{belongsTo}(L,S),\ \mathit{not\ holds}((L,0)).$

(PEC14) $\bot \leftarrow \mathit{instant}(I),\ \mathit{effectChoice}((X,P),I),$
$\mathit{fluent}(F),\ \mathit{belongsTo}((F,V),X),$
$\mathit{not\ holds}(((F,V),I+1)),\ I < \mathit{maxinst}.$

(PEC15) $\bot \leftarrow \mathit{instant}(I),\ \mathit{fluent}(F),\ \mathit{not\ holds}(((F,V),I)),$
$\mathit{effectChoice}((X,P),I),\ \mathit{not\ belongsTo}((F,V),X),$
$\mathit{holds}(((F,V),I+1)),\ I < \mathit{maxinst}.$

(PEC16) $\bot \leftarrow \mathit{fluent}(F),\ \mathit{instant}(I),\ \mathit{holds}(((F,V),I)),$
$\mathit{not\ inOcc}(I),\ \mathit{not\ holds}(((F,V),I+1)),$
$I < \mathit{maxinst}.$

(PEC17) $\mathit{eval}(A,I,P) \leftarrow \mathit{action}(A),\ \mathit{instant}(I),$
$\mathit{performed}(A,I,P),\ \mathit{holds}(((A,\mathit{true}),I)).$

(PEC18) $\mathit{eval}(A,I,1-P) \leftarrow \mathit{action}(A),\ \mathit{instant}(I),$
$\mathit{performed}(A,I,P),\ \mathit{holds}(((A,\mathit{false}),I)).$

Informally, axiom *PEC1* states that all actions are boolean (see definition 3.1); axiom *PEC2* defines a characteristic predicate for $\mathcal{F} \cup \mathcal{A}$; axioms *PEC3* and *PEC4* define literals and i-literals, respectively. Axioms *PEC5* and *PEC6* define the two auxiliary predicates *definitelyPerformed* and *possiblyPerformed* representing the sets of actions and instants such that A is certainly

performed at I (i.e., with probability 1) and such that A might have been performed at I (i.e., with a probability greater than 0) respectively.

Axioms *PEC7* to *PEC18* correspond to the definitions introduced in the previous section, namely: axiom *PEC7* corresponds to definition 3.14, axiom *PEC8* defines a characteristic predicate for $occ_{\mathcal{D}}$ as in definition 3.17, axioms *PEC9* and *PEC14* to *PEC16* correspond to justified change, axioms *PEC10* and *PEC13* corresponds to the initial condition, axioms *PEC11* and *PEC12* correspond to CWA for actions, axioms *PEC17* and *PEC18* implement eq. (3.10).

We denote the domain-independent part of our theory, i.e. axioms *PEC1* to *PEC18*, by Π_I. Notice that axioms *PEC11* to *PEC16* are constraints, and in the following will be referred to as Π_C.

4.3 Correctness

We now show that the provided translation is sound and complete with respect to the definitions given in chapter 3, in the sense that there is a one-to-one correspondence between the traces of a domain description and the answer sets of its corresponding ASP program. This proof relies on the Splitting Theorem [41], a useful tool to obtain the answer sets of a ground program.

4.3.1 Splitting Set Theorem

A set U of ground atoms is a *splitting set* for a ground program Π if, for each rule in Π, if U contains some atom in the head of the rule, then it also contains all the atoms occurring in that rule. For instance, if $\Pi' = \{a \leftarrow \mathit{not}\ b, b \leftarrow c, c\}$ then $\{a, b, c\}$, $\emptyset$, $\{b, c\}$ and $\{c\}$ are splitting sets for Π', whereas $\{a, b\}$, $\{a\}$ and $\{b\}$ are not.

A splitting set U splits an answer set program Π into a bottom program $\mathit{bot}_U(\Pi)$ and a top program $\mathit{top}_U(\Pi) = \Pi \setminus \mathit{bot}_U(\Pi)$. With the program Π' defined as above, $U' = \{c\}$ splits Π' into $\mathit{bot}_{U'}(\Pi') = \{c\}$ and $\mathit{top}_{U'}(\Pi') = \{a \leftarrow \mathit{not}\ b, b \leftarrow c\}$. So $\mathit{bot}_U(\Pi)$ contains all the rules of Π with a member of U as their head.

The *splitting set theorem* states that the answer sets of Π are exactly those that can be expressed as $X \cup Y$ for some X that is an answer set of $\mathit{bot}_U(\Pi)$ and some Y that is an answer set of $e_U(\mathit{top}_U(\Pi), X)$. The set $e_U(\mathit{top}_U(\Pi), X)$ denotes the *partial evaluation of the program* $\mathit{top}_U(\Pi)$ *w.r.t.* U defined as follows: a rule r is in $e_U(\mathit{top}_U(\Pi), X)$ if and only if there exists a rule $r' \in \mathit{top}_U(\Pi)$ such that (i) all positive literals that are both in the body of r' and in U are also in X, (ii) there is no negative literal in the body of r' whose corresponding atom is in both U and X, and (iii) the rule r is obtained from r' by removing all literals from the body of r' whose corresponding atom is in U. If we consider Π' and U' again and let X' be the only answer set $\{c\}$ of $\mathit{bot}_{U'}(\Pi') = \{c\}$, $\Pi'' = e_{U'}(\mathit{top}_{U'}(\Pi'), X') = \{a \leftarrow \mathit{not}\ b, b\}$ and notice that now we can split Π'' itself. If we let $U'' = \{b\}$, then $\mathit{bot}_{U''}(\Pi'') = \{b\}$ and $\Pi''' = e_{U''}(\mathit{top}_{U''}(\Pi''), X'') = \emptyset$ for the only answer set $X'' = \{b\}$ of $\mathit{bot}_{U''}(\Pi'')$. The answer sets of the original program Π' can now be obtained as $X' \cup X'' \cup X'''$, where $X' = \{c\}$ is the answer set of $\mathit{bot}_{U'}(\Pi')$, $X'' = \{b\}$ is the answer set of $\mathit{bot}_{U''}(\Pi'') = \{c\}$ and $X''' = \emptyset$ is the answer set of Π'''. Then, the program Π' has only one answer set $\{b, c\}$.

```
instant(0) instant(1) instant(2) fluent(coin) possVal(coin,heads)
possVal(coin,tails) action(toss) possVal(toss,true) possVal(toss,false)
belongsTo((coin,heads),id0) belongsTo((coin,tails),id2)
belongsTo((coin,heads),id3) literal((coin,heads))
literal((coin,tails)) literal((toss,true)) literal((toss,false))
iliteral(((coin,heads),0)) iliteral(((coin,tails),0))
iliteral(((toss,true),0)) iliteral(((toss,false),0))
iliteral(((coin,heads),1)) iliteral(((coin,tails),1))
iliteral(((toss,true),1)) iliteral(((toss,false),1))
iliteral(((coin,heads),2)) iliteral(((coin,tails),2))
iliteral(((toss,true),2)) iliteral(((toss,false),2))
initialCondition((id0,1)) holds(((coin,heads),0))
holds(((toss,false),0)) holds(((coin,heads),1)) holds(((toss,true),1))
holds(((coin,heads),2)) holds(((toss,false),2)) performed(toss,1,1)
initialChoice((id0,1)) possiblyPerformed(toss,1)
definitelyPerformed(toss,1) fluentOrAction(coin) fluentOrAction(toss)
causedOutcome((id1,2/100),1) causedOutcome((id2,49/100),1)
causedOutcome((id3,49/100),1) inOcc(1) effectChoice((id3,49/100),1)
eval(toss,1,1)
```

Figure 4.1: An answer set of $\Pi_{\mathcal{D}_C} \cup \Pi_I$, with $\mathcal{D}_C$ defined as in example 3.6. This answer set corresponds to the trace $\langle(\mathit{Coin}{=}\mathit{Heads}, 1)@\bar{\mathbb{X}}, (\mathit{Coin}{=}\mathit{Heads}, 0.49)@1\rangle$.

Finally, a remark about notation: if the considered language includes predicate symbols, splitting a program Π with respect to some predicates $p_1, \ldots, p_n$ means splitting it with respect to the set U of all groundings of $p_1, \ldots, p_n$, and so in this context $U = \{p_1, \ldots, p_n\}$ signifies this set of all groundings.

4.3.2 Basic Properties of Stable Models

In the following, we will use the fact that answer sets of a program consisting only of the choice rule $\{a_1, \ldots, a_n\}$ are the power set $\{\emptyset, \{a_1\}, \ldots, \{a_n\}, \{a_1, a_2\}, \ldots, \{a_1, \ldots, a_n\}\}$, and that answer sets of a constrained choice rule $X\{a_1, \ldots, a_n\}Y$ are the answer sets of $\{a_1, \ldots, a_n\}$ with cardinality $\geq X$ and $\leq Y$. Also, we use the fact that the only answer set of the program $\{p(X) : q(X), q(a_1), \ldots, q(a_n)\}$, where $p(X) : q(X)$ is called a conditional literal, is $\{p(a_1), \ldots, p(a_n), q(a_1), \ldots, q(a_n)\}$. Notice that conditional literal and choice rules can be combined so that e.g. answer sets of the program $\{q(a, b), q(a, c), q(b, c), 1\{p(X) : q(a, X)\}1\}$ are $\{q(a, b), q(a, c), q(b, c), p(b)\}$ and $\{q(a, b), q(a, c), q(b, c), p(c)\}$. Finally, constraints are used to eliminate answer sets that satisfy its body, e.g. answer sets of the program $\{q(a, b), q(a, c), q(b, c), 1\{p(X) : q(a, X)\}1, \bot \leftarrow p(b)\}$ are the answer sets of $\{q(a, b), q(a, c), q(b, c), 1\{p(X) : q(a, X)\}1\}$ that do not satisfy $p(b)$, hence $\{q(a, b), q(a, c), q(b, c), p(c)\}$ is its only answer set.

Since the splitting set theorem can only be applied to ground programs, in the following we will interpret non-ground axioms as shorthand for the set of all their ground instances, e.g. the axiom $p(X) \leftarrow q(X, Y)$ from the program $\{p(X) \leftarrow q(X, Y), q(a, b)\}$ is shorthand for the set $\{p(a) \leftarrow q(a, a), p(a) \leftarrow q(a, b), p(b) \leftarrow q(b, a), p(b) \leftarrow q(b, b)\}$.

4.3.3 Stable Models of $\Pi_I \cup \Pi_{\mathcal{D}}$

This section derives a characterisation of the stable models of the program $\Pi_{\mathcal{D}} \cup \Pi_I$.

Let Π be the program obtained by grounding $\Pi_{\mathcal{D}} \cup \Pi_I \setminus \Pi_C$. We split Π with respect to $U = \{\textit{fluent}, \textit{action}, \textit{instant}, \textit{possVal}\}$. The bottom $bot_U(\Pi)$ is guaranteed by the translation process to have a unique answer set $Z_{\mathcal{L}}$ which includes a correct representation of the domain language $\mathcal{L}$, i.e. of the three sorts $\mathcal{F}$, $\mathcal{A}$ and $\mathcal{I}$ and of the function *vals* (note that the definition of $\mathcal{V}$ is implicitly derived from that of *vals* and that our implementation is restricted to the case where $\leq = \leq_{\mathbb{N}}$ and $\bar{0} = 0$), meaning that:

$$\textit{fluent}(f) \in Z_{\mathcal{L}} \Leftrightarrow F \in \mathcal{F}$$
$$\textit{action}(a) \in Z_{\mathcal{L}} \Leftrightarrow A \in \mathcal{A}$$
$$\textit{instant}(i) \in Z_{\mathcal{L}} \Leftrightarrow I \in \mathcal{I}$$
$$\textit{possVal}(f, v) \in Z_{\mathcal{L}} \Leftrightarrow F \in \mathcal{F} \cup \mathcal{A}, V \in \textit{vals}(F)$$

Let $\Pi_0 = e_U(top_U(\Pi), Z_{\mathcal{L}})$ and split it using the set $U_0 = \{\textit{fluentOrAction}, \textit{literal}, \textit{iLiteral}\}$. The bottom $bot_{U_0}(\Pi_0)$ consists of axioms *PEC2* to *PEC4*. Evaluating it w.r.t. $Z_{\mathcal{L}}$ gives that it has answer set Z_A characterised as follows:

$$\textit{fluentOrAction}(x) \in Z_A \Leftrightarrow \textit{fluent}(x) \in Z_{\mathcal{L}} \vee \textit{action}(x) \in Z_{\mathcal{L}} \Leftrightarrow X \in \mathcal{F} \cup \mathcal{A}$$
$$\textit{literal}((x, v)) \in Z_A \Leftrightarrow \textit{possVal}(x, v) \in Z_{\mathcal{L}} \Leftrightarrow X \in \mathcal{F} \cup \mathcal{A}, V \in \textit{vals}(X)$$
$$\textit{iLiteral}((x, v)) \in Z_A \Leftrightarrow \textit{possVal}(x, v) \in Z_{\mathcal{L}} \wedge \textit{instant}(i) \in Z_{\mathcal{L}} \Leftrightarrow X \in \mathcal{F} \cup \mathcal{A}, V \in \textit{vals}(X), I \in \mathcal{I}$$

We now split the partially evaluated top $\Pi_1 = e_{U_0}(top_U(\Pi_0), Z_A)$ using the set $U_1 = \{\textit{holds}\}$. The bottom $bot_{U_1}(\Pi_1)$ consists only of axiom *PEC7* and has answer sets that correspond to any possible world in the domain language, i.e., according to the semantics of choice rules axiom *PEC7* generates every possible function from instants to states, hence for a particular world $W \in \mathcal{W}$ we denote by Z_W the corresponding answer set of $bot_{U_1}(\Pi_1)$ which is characterised by

$$\textit{holds}(((x, v), i)) \in Z_W \Leftrightarrow W \mid\!\models [X = V]@I$$

Notice that for any fixed $W \in \mathcal{W}$ the three sets of propositions

$$\Pi_2 = e_{U_1}(\{\textit{PEC8}, \textit{PEC9}\} \cup C_{\mathcal{D}}, Z_W)$$
$$\Pi_3 = e_{U_1}(\{\textit{PEC10}\} \cup I_{\mathcal{D}}, Z_W)$$
$$\Pi_4 = e_{U_1}(\{\textit{PEC5}, \textit{PEC6}, \textit{PEC17}, \textit{PEC18}\} \cup P_{\mathcal{D}}, Z_W)$$

partition $e_{U_1}(top_{U_1}(\Pi_1), Z_W)$ and are independent of each other, so we can evaluate their answer sets separately.

We start with Π_2 and split it with the set $U_2 = \{\textit{causedOutcome}, \textit{belongsTo}\}$. The translation procedure guarantees that its bottom $bot_{U_2}(\Pi_2)$ has answer set C_W such that

$$\textit{belongsTo}((x, v), id_j^n) \in C_W \Leftrightarrow c_n \in \mathcal{D}, O_j \in \textit{body}(c_n), X = V \in \chi(O_j)$$

and

$$\textit{causedOutcome}((id_j^n, p), i) \in C_W \Leftrightarrow$$
$$c_n \in \mathcal{D}, O_j \in \textit{body}(c_n), \textit{holds}([\textit{head}(c_n)]@I) \in Z_W, P = \pi(O_j) \Leftrightarrow$$
$$c_n \in \mathcal{D}, O_j \in \textit{body}(c_n), W \mid\!\models [\textit{head}(c_n)]@I, P = \pi(O_j) \Leftrightarrow$$

$instant(0)$ $instant(1)$	$fluentOrAction(coin)$	
$instant(2)$ $fluent(coin)$	$fluentOrAction(toss)$	
$action(toss)$	$literal((coin, heads))$	
$possVal(coin, heads)$	$literal((coin, tails))$	$holds(((coin, heads), 0))$
$possVal(coin, tails)$	$literal((toss, true))$	$holds(((toss, false), 0))$
$possVal(toss, true)$	$literal((toss, false))$	$holds(((coin, heads), 1))$
$possVal(toss, false)$	$iLiteral(((coin, heads), 0))$	$holds(((toss, true), 1))$
	$\vdots$	$holds(((coin, heads), 2))$
	$iLiteral(((toss, false), 2))$	$holds(((toss, false), 2))$
$Z_{\mathcal{L}}$ *ans. set of* $bot_U(\Pi)$	Z_A *ans. set of* $bot_{U_0}(\Pi_0)$	Z_W *ans. set of* $bot_{U_1}(\Pi_1)$

$belongsTo((coin, tails), id_2)$		
$belongsTo((coin, heads), id_3)$		
$causedOutcome((id_1, 2/100), 1)$		
$causedOutcome((id_2, 49/100), 1)$		$performed(toss, 1, 1)$
$causedOutcome((id_3, 49/100), 1)$	$belongsTo((coin, heads), id_0)$	$possiblyPerformed(toss, 1)$
$inOcc(1)$	$initialCondition((id_0, 1))$	$definitelyPerformed(toss, 1)$
$effectChoice((id_3, 49/100), 1)$	$initialChoice((id_0, 1))$	$eval(toss, 1, 1)$
$C_W \cup O_W \cup E_{ec}$ *ans. set of* Π_2	$I_{\mathcal{D}} \cup I_{ic}$ *ans. set of* Π_3	$P_W \cup P_W^A \cup Ev_W$ *ans. set of* Π_4

Figure 4.2: The answer set in fig. 4.1 split into its components.

where $c_n \in \mathcal{D}$ is the nth c-proposition in the enumeration fixed during the translation process (see section 4.1 for reference).

Now consider the partially evaluated top $\Pi_2^1 = e_{U_2}(\Pi_2, C_W)$ and split it using $U_2^1 = \{inOcc\}$. The bottom program $bot_{U_2^1}(\Pi_2^1)$ only consists of axiom *PEC8* and its answer set O_W is characterised as follows:

$$inOcc(i) \in O_W \Leftrightarrow \exists o : i \in Z_{\mathcal{L}}, causedOutcome(o, i) \in C_W \Leftrightarrow$$
$$\text{a cause occurs at } I \text{ in } W \text{ w.r.t. } \mathcal{D} \Leftrightarrow I \in occ_{\mathcal{D}}(W)$$

We now need to evaluate and find the answer sets of $\Pi_2^2 = e_{U_2^1}(top_{U_2^1}(\Pi_2^1), O_W)$ which now consists only of a partially evaluated axiom *PEC9*.

The role of axiom *PEC9* is to implement the *effectChoice* function. Indeed, for each instant I such that $I \in occ_{\mathcal{D}}(W)$, exactly one atom of the form $effectChoice(o, i)$ is included in an answer set of Π_2^2 for some o such that $causedOutcome(o, i) \in C_W$. Since this is consistent with definition 3.19, $effectChoice(o, i)$ correctly represents its intended semantic counterpart $ec(I) = O$ where *ec* is an effect choice function for W w.r.t. $\mathcal{D}$. For an effect choice function *ec* for W w.r.t. $\mathcal{D}$, we call the corresponding answer set that encodes it E_{ec}.

Applying the splitting theorem, we can now conclude that answer sets of Π_2 are exactly those given by the set $\{C_W \cup O_W \cup E_{ec} \mid ec \text{ is an effect choice function for } W \text{ w.r.t. } \mathcal{D}\}$.

Answer sets of Π_3 correspond to the *initialChoice* constant and can be worked out in a similar way as in the effect choice function case. It can be shown that $initialChoice(o)$ correctly represents an initial choice *ic* as in definition 3.20, and answer sets of Π_3 are given by $\{I_{\mathcal{D}} \cup I_{ic} \mid ic \text{ is an initial choice w.r.t. } \mathcal{D}\}$ where I_{ic} encodes *ic*.

Finally we need to derive answer sets of Π_4. We split it using $U_4 = \{\textit{performed}\}$. The bottom $bot_{U_4}(\Pi_4)$ has answer set P_W consisting of atoms of the form $\textit{performed}(a, i, p^+)$ for each p-proposition "A **performed-at** I **with-prob** P^+ **if-holds** θ" in $\mathcal{D}$ such that $W \models [\theta]@I$.

Let $\Pi_4^1 = e_{U_4}(top_{U_4}(\Pi_4, P_W), P_W)$ and split it using $U_4^1 = \{\textit{possiblyPerformed}, \textit{definitelyPerformed}\}$. The bottom $bot_{U_4^1}(\Pi_4^1)$ has an answer set P_W^A characterised as follows:

$$\textit{possiblyPerformed}(a, i) \in P_W^A \Leftrightarrow \exists p^+, \textit{performed}(a, i, p^+) \in P_W \Leftrightarrow$$
$$\exists \theta, P^+, \text{"}A \textbf{ performed-at } I \textbf{ with-prob } P^+ \textbf{ if-holds } \theta\text{"} \in \mathcal{D} \text{ and } W \models [\theta]@I$$

and

$$\textit{definitelyPerformed}(a, i) \in P_W^A \Leftrightarrow \exists p^+, \textit{performed}(a, i, 1) \in P_W \Leftrightarrow$$
$$\exists \theta, \text{"}A \textbf{ performed-at } I \textbf{ with-prob } 1 \textbf{ if-holds } \theta\text{"} \in \mathcal{D} \text{ and } W \models [\theta]@I$$

We are now left with calculating answer sets of $\Pi_4^2 = e_{U_4^1}(top_{U_4^1}(\Pi_4^1, P_W^A), P_W^A)$. The aim of *eval* is that of implementing eq. (3.10). It is important to notice here that, thanks to requirement iv) in definition 3.11, it is possible to label a p-proposition "A **performed-at** I **with-prob** P^+ **if-holds** θ" using only A and I. Comparing eq. (3.10) with axioms *PEC17* and *PEC18* immediately gives that the only answer set of Π_4 is $P_W \cup P_W^A \cup Ev_W$ where

$$Ev_W = \Big\{ \textit{eval}(a, i, p) \mid A \in \mathcal{A}, I \in \mathcal{I}, \text{"}A \textbf{ performed-at } I \textbf{ with-prob } P^+ \textbf{ if-holds } \theta\text{"} \in \mathcal{D},$$
$$P = P^+ \text{ if } W \models [A]@I, P = 1 - P^+ \text{ otherwise} \Big\}.$$

We are now able to calculate the answer sets of the whole program $\Pi \setminus \Pi_C$, which are given by the set:

$$\Big\{ Z_{\mathcal{L}} \cup Z_A \cup Z_W \cup C_W \cup O_W \cup E_{ec} \cup I_{\mathcal{D}} \cup I_{ic} \cup P_W \cup P_W^A \cup Ev_W \mid W \in \mathcal{W}, \textit{ec} \text{ is an effect}$$
$$\text{choice for } W \text{ w.r.t. } \mathcal{D} \text{ and } \textit{ic} \text{ is an initial choice w.r.t. } \mathcal{D} \Big\}$$

and we write $Z(\mathcal{L}, \mathcal{D}, W, tr)$ for it, where $tr = \langle \textit{ic}, \textit{ec} \rangle$.

Finally, we take into account the constraints Π_C, whose effect is that of implementing the Closed World Assumption and the effects of initialisation and persistence. Since axioms *PEC11* to *PEC16* are constraints, they eliminate those answer sets of Π that satisfy their bodies.

Let Z be an answer set of Π of the form $Z(\mathcal{L}, \mathcal{D}, W, tr)$. Axioms *PEC11* and *PEC12* together with the correctness of translation of p-proposition's preconditions ensure that:

$$W \models [A]@I \Leftrightarrow \textit{holds}(((a, \textit{true}), i)) \in Z \Rightarrow \exists p, \textit{performed}(a, i, p^+) \in \Pi_{\mathcal{D}} \Leftrightarrow$$
$$\exists \theta, \text{"}A \textbf{ performed-at } I \textbf{ with-prob } P^+ \textbf{ if-holds } \theta\text{"} \in \mathcal{D} \text{ and } W \models [\theta]@I$$

and, conversely

$$W \models [\neg A]@I \Leftrightarrow \textit{holds}(((a, \textit{false}), i)) \in Z \Rightarrow \textit{performed}(a, i, 1) \notin \Pi_{\mathcal{D}} \Leftrightarrow$$
$$\nexists \theta, \text{"}A \textbf{ performed-at } I \textbf{ with-prob } 1 \textbf{ if-holds } \theta\text{"} \notin \mathcal{D} \text{ and } W \models [\theta]@I$$

therefore the world encoded in Z_W must satisfy CWA, i.e. definition 3.16.

Let now "**initially-one-of** $(O_1, O_2, \ldots, O_m)$" be an i-proposition in $\mathcal{D}$ and Z_W be as before. Axiom *PEC14* makes sure that:

$$\textit{holds}(((f,v),0)) \in Z_W \Leftrightarrow \exists s : \{\textit{initialChoice}((s,p)), \textit{belongsTo}((f,v),s)\} \subseteq \Pi_{\mathcal{D}} \Leftrightarrow$$
$$\exists O : O \in \{O_1,\ldots,O_m\}, \tilde{S} = \chi(O), [F=V] \in \tilde{S}.$$

which satisfies the initial condition, i.e. definition 3.18.

Finally we consider axioms *PEC14* to *PEC16*. Let I, I' be two instants with $I < I'$ as in definition 3.23, consider the world encoded in Z_W and let $W(I)\restriction\mathcal{F} = \tilde{S}$ and $W(I')\restriction\mathcal{F} = \tilde{S}'$. Assume that the *effectChoice* function encoded in Z maps instants in $occ_{\mathcal{D}}(W) \cap [I, I')$ to outcomes $O_1, O_2, \ldots, O_n$. Axiom *PEC16* makes sure that $\tilde{S}$ cannot be altered if $I \notin \textit{inOcc}_{\mathcal{D}}(W)$. Therefore $\tilde{S}$ can only change at instants $I \in occ_{\mathcal{D}}(W)$. We now show that $\tilde{S}'$ is actually equal to $\tilde{S} \oplus O_1 \oplus O_2 \oplus \cdots \oplus O_n$. If not, and considering that our implementation is restricted to a finite set of instants, either (i) there is a fluent literal $L \in \chi(O)$ for some $O \in \{O_1, O_2, \ldots, O_n\}$ and an instant $I'' \in [I, I')$ such that $L \in \chi(O)$ but $L \notin W(I''+1)$, or (ii) for some $O \in \{O_1, O_2, \ldots, O_n\}$ and a fluent literal $L = F{=}V$ such that $L \notin O$ and $L \notin W(I'')$, $L \in W(I''+1)$. Both (i) and (ii) are forbidden by axioms *PEC15* and *PEC16* respectively, by considering that the answer set Z correctly represents the semantic objects that it encodes.

4.3.4 Correctness Statement and Proof

Before proving the correctness of the implementation, we need to define a correspondence between answer sets and traces. This is the aim of the following definitions:

Definition 4.1 (Manifest Choice Element)**.** We say that the *choice element* $(\tilde{X}, P^+)@I$ is *manifest in the answer set* Z if and only if there exists a symbol *id* such that $\textit{effectChoice}((\textit{id}, p^+), i) \in Z$ and such that $L \in \tilde{X}$ if and only if $\textit{belongsTo}(l, \textit{id}) \in Z$ (recall that p^+, i and l are the ASP representations of P^+, I and L respectively).

Definition 4.2 (Trace of an answer set)**.** Let Z be an answer set of $\Pi_{\mathcal{D}} \cup \Pi_I$. If Z is such that for each $I \in \mathcal{I}$ there is at most one choice element $(X, P^+)@I$ that is manifest in Z, then the *trace of* Z is the trace $\langle O_{\bigtriangledown}@\bigtriangledown, O_1@I_1, \ldots, O_n@I_n\rangle$ where $O_{\bigtriangledown}@\bigtriangledown, O_1@I_1, \ldots, O_n@I_n$ are exactly the manifest choice elements in Z ordered according to instants $I_1, \ldots, I_n$. Otherwise, the trace of Z is undefined.

Proposition 4.1. A candidate trace tr is a trace of $\mathcal{D}$ if and only if there exists an answer set Z of $\Pi_{\mathcal{D}} \cup \Pi_I$ such that tr is a trace of Z.

Proof. It follows from the commentary in section 4.3.3 by considering that Z is an answer set of $\Pi_{\mathcal{D}} \cup \Pi_I$ if and only if it has the form $Z(\mathcal{L}, \mathcal{D}, W, tr)$ for W a well-behaved world w.r.t. $\mathcal{D}$ and a trace tr of W w.r.t. $\mathcal{D}$. Therefore, if tr is a trace of $\mathcal{D}$ by definition it is possible to find a world W that is well-behaved w.r.t. $\mathcal{D}$ and has trace W, then just consider $Z(\mathcal{L}, \mathcal{D}, W, tr)$ for an answer set of $\Pi_I \cup \Pi_{\mathcal{D}}$ such that tr is a trace of it. Conversely, if Z is an answer set of $\Pi_I \cup \Pi_{\mathcal{D}}$ and tr is its trace, then there exists some world $W \in \mathcal{W}$ well-behaved w.r.t. $\mathcal{D}$ such that W has trace tr w.r.t. $\mathcal{D}$, and therefore tr is a trace of $\mathcal{D}$. □

4.4 Implementation

PEC+ has been implemented for the class of domains in which the bodies of c-propositions θ are conjunctive formulas. The implementation and example domain descriptions can be found

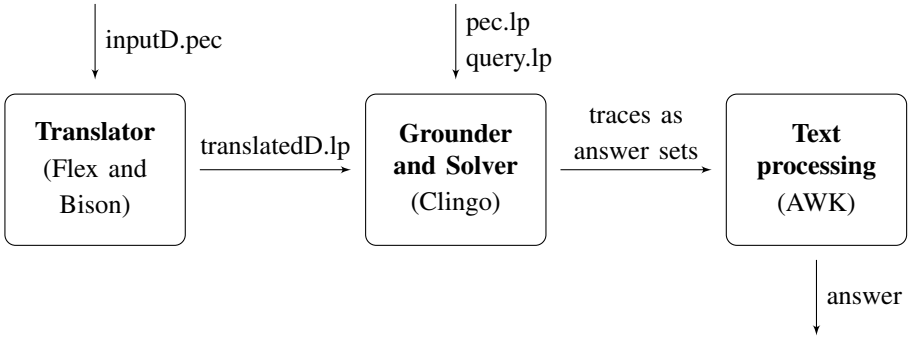


Figure 4.3: Schematic representation of the implementation of PEC+.

on GitHub at `https://github.com/dasaro/pec`. An overview of the implementation is given in fig. 4.3. A translator turns a PEC+ domain description $\mathcal{D}$ into an ASP program using the lexical analyser Flex and the parser generator Bison (see e.g. [40]). The output of the translator together with an ASP formulation of the domain-independent part of the semantics and a query[1] are then processed by Clingo (see e.g. [25]). This returns answer sets, each of which represents a trace of the domain description which is compatible with the query (as discussed in the introduction of this chapter) and the corresponding well-behaved world w.r.t. $\mathcal{D}$. A standard text processing tool, AWK (see e.g. [2]) then processes the resulting answer sets and calculates the probability associated with the given query using eq. (3.14).

This section briefly presents some empirical results about this implementation and are summarised in table 4.1 and figs. 4.4 to 4.6. All the experiments in this chapter were run on an Mid-2010 Apple MacBook Core 2 Duo 2.4 GHz.

Table 4.1 and fig. 4.4 are an application of the implementation to scenario 1.3 and show how it can handle some real-world domains (computation time never exceeded 1 second for all queries).

Figure 4.4b shows that there is a very high Pearson correlation (> 0.99) between computation time and number of traces, which is confirmed in the following experiment:

Example 4.5 (Decay)**.** Let the set of instants for this example be $\{0, \ldots, 15\}$ and consider a domain description $\mathcal{D}_D$ consisting of the following propositions:

(D1) F **takes-values** $\{\top, \bot\}$

(D2) **initially-one-of** $\{(\{F = \top\}, 1)\}$

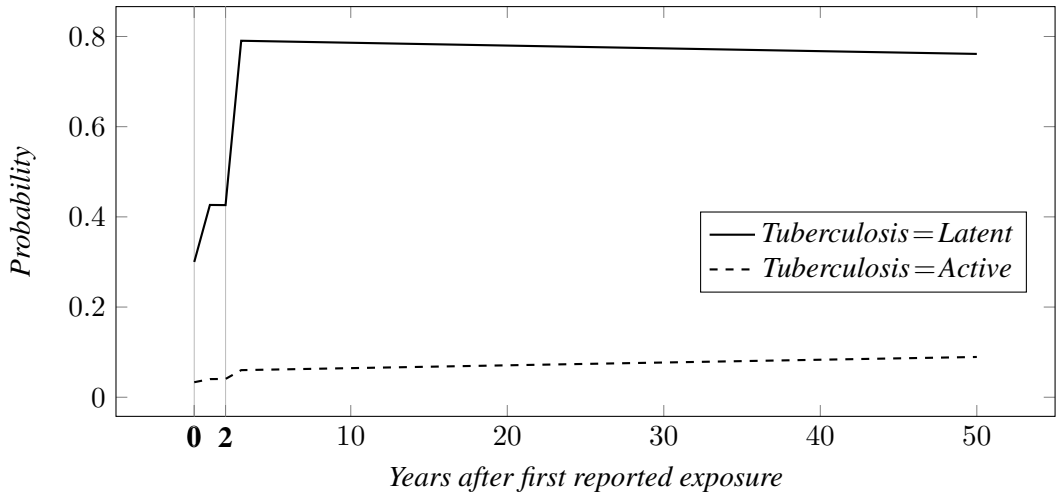
(D3) A **causes-one-of** $\{(\{F = \bot\}, 1/5)\}$

(D4) A **performed-at** I **with-prob** $1/2$

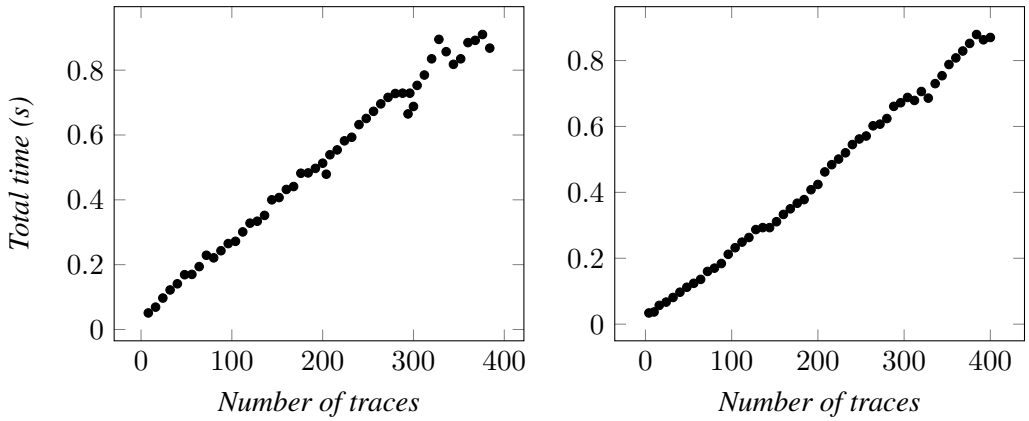
and the two collections of queries $[F = \top]@I$ and $[F = \bot]@I$ for $I = 0, \ldots, 15$.

The essential feature of this example is that by varying I in the query, between 0 and 15, one can vary the number of traces of the domain description consistent with the query, and therefore the number of answer sets generated. As shown in fig. 4.5, the Pearson correlation between

[1] A mechanism for translating arbitrary queries to ASP programs is not available yet but it is planned for future work.



(a) Probability of Tuberculosis = Latent and Tuberculosis = Active at different instants. The two (probabilistic) exposure events happening at instants 0 *and* 2 *make the probabilities increase. Possible reactivation of the pathogen (see axiom schema* T5*) makes the probability of Tuberculosis = Latent decay and that of Tuberculosis = Active increase at every instant.*



(b) Scatter plots showing the relation between number of traces and total execution time. The plot on the left refers to the computation of answers to the query "$[$*Tuberculosis = Latent*$]@I$*" while the plot on the left refers to the query "*$[$*Tuberculosis = Active*$]@I$*" for* $I \in \{0, 1, \ldots, 50\}$*. Correlation between number of traces and total execution time is* ≈ 0.9946 *for the plot on the left and* ≈ 0.9983 *for the plot on the right.*

Figure 4.4

number of traces and total execution time is again very high (>0.99). In this example, average execution time is much higher, almost reaching 1 hour for some queries. This is due to (unconditional) probabilistic actions which cause the number of traces to grow exponentially. This problem can be somewhat circumvented if we consider that PEC+ satisfies the causality principle (see proposition 3.3) and therefore a truncated domain description $\mathcal{D}_{<I}$ can be considered when answering queries mentioning at most instant I. This causes dramatic improvements in some cases, as shown in fig. 4.6.

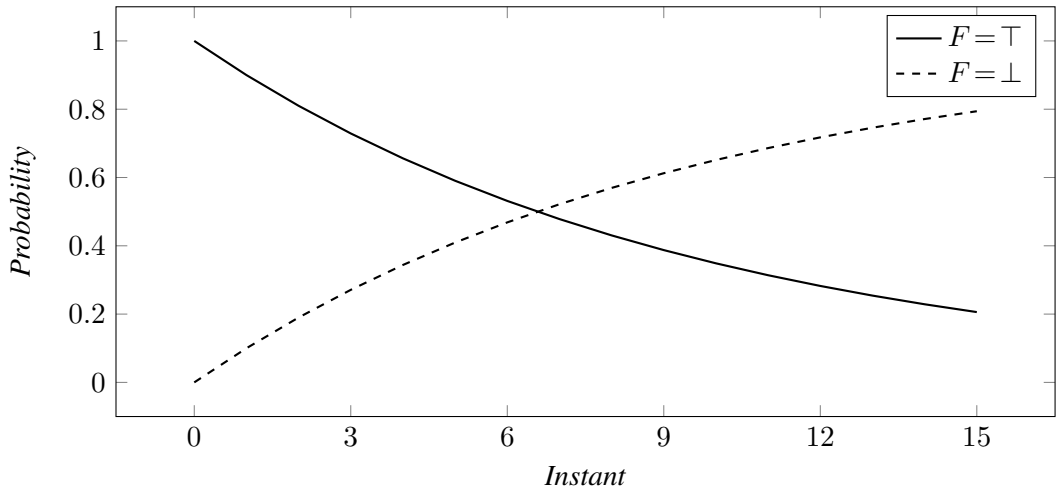
Finally, it can be immediately seen from the execution times that *Total Time*$-$*Solving Time*$\approx$*0*, meaning that the time spent on preprocessing and grounding is negligible.

Query:	$[Tuberculosis = Latent]@I$ **holds-with-prob** P				$[Tuberculosis = Active]@I$ **holds-with-prob** P			
I	P	n Traces	T Time	S Time	P	n Traces	T Time	S Time
0	0.3	204	0.479	0.44	0.0333333	4	0.034	0
1	0.426427	300	0.688	0.66	0.04024	10	0.037	0.01
2	0.426086	294	0.665	0.63	0.0405811	16	0.057	0.02
3	0.790545	384	0.868	0.84	0.060122	24	0.067	0.03
4	0.789912	376	0.91	0.87	0.0607544	32	0.081	0.05
5	0.78928	368	0.892	0.86	0.0613864	40	0.097	0.06
6	0.788649	360	0.885	0.85	0.0620178	48	0.112	0.08
7	0.788018	352	0.835	0.8	0.0626487	56	0.124	0.09
8	0.787388	344	0.818	0.79	0.0632791	64	0.136	0.11
9	0.786758	336	0.857	0.83	0.063909	72	0.16	0.13
10	0.786128	328	0.895	0.86	0.0645384	80	0.17	0.14
11	0.785499	320	0.835	0.8	0.0651674	88	0.184	0.15
12	0.784871	312	0.785	0.74	0.0657958	96	0.212	0.18
13	0.784243	304	0.753	0.72	0.0664236	104	0.232	0.2
14	0.783616	296	0.729	0.7	0.067051	112	0.249	0.21
15	0.782989	288	0.729	0.7	0.0676779	120	0.263	0.23
16	0.782362	280	0.728	0.7	0.0683043	128	0.287	0.25
17	0.781736	272	0.716	0.68	0.0689302	136	0.293	0.25
18	0.781111	264	0.696	0.66	0.0695556	144	0.293	0.26
19	0.780486	256	0.673	0.64	0.0701805	152	0.311	0.28
20	0.779862	248	0.651	0.62	0.0708049	160	0.333	0.3
21	0.779238	240	0.632	0.59	0.0714288	168	0.35	0.32
22	0.778615	232	0.593	0.56	0.0720522	176	0.367	0.34
23	0.777992	224	0.582	0.55	0.0726751	184	0.378	0.35
24	0.777369	216	0.554	0.52	0.0732974	192	0.408	0.38
25	0.776747	208	0.539	0.51	0.0739193	200	0.424	0.39
26	0.776126	200	0.513	0.48	0.0745407	208	0.462	0.43
27	0.775505	192	0.497	0.47	0.0751616	216	0.484	0.45
28	0.774885	184	0.483	0.45	0.075782	224	0.501	0.46
29	0.774265	176	0.482	0.45	0.076402	232	0.52	0.48
30	0.773645	168	0.441	0.41	0.0770214	240	0.545	0.51
31	0.773026	160	0.432	0.4	0.0776403	248	0.562	0.53
32	0.772408	152	0.407	0.38	0.0782587	256	0.571	0.54
33	0.77179	144	0.4	0.37	0.0788766	264	0.602	0.57
34	0.771173	136	0.352	0.32	0.0794941	272	0.607	0.58
35	0.770556	128	0.334	0.3	0.080111	280	0.624	0.6
36	0.769939	120	0.328	0.29	0.0807274	288	0.661	0.62
37	0.769323	112	0.301	0.27	0.0813434	296	0.672	0.63
38	0.768708	104	0.272	0.24	0.0819589	304	0.688	0.65
39	0.768093	96	0.265	0.23	0.0825738	312	0.679	0.65
40	0.767478	88	0.243	0.21	0.0831883	320	0.706	0.68
41	0.766864	80	0.221	0.19	0.0838023	328	0.686	0.66
42	0.766251	72	0.229	0.19	0.0844158	336	0.73	0.7
43	0.765638	64	0.194	0.16	0.0850288	344	0.754	0.72
44	0.765025	56	0.17	0.14	0.0856413	352	0.788	0.75
45	0.764413	48	0.169	0.13	0.0862533	360	0.808	0.77
46	0.763802	40	0.141	0.11	0.0868648	368	0.829	0.79
47	0.763191	32	0.122	0.08	0.0874759	376	0.852	0.82
48	0.76258	24	0.097	0.06	0.0880864	384	0.879	0.84
49	0.76197	16	0.069	0.04	0.0886965	392	0.863	0.83
50	0.761361	8	0.051	0.02	0.0893061	400	0.87	0.84

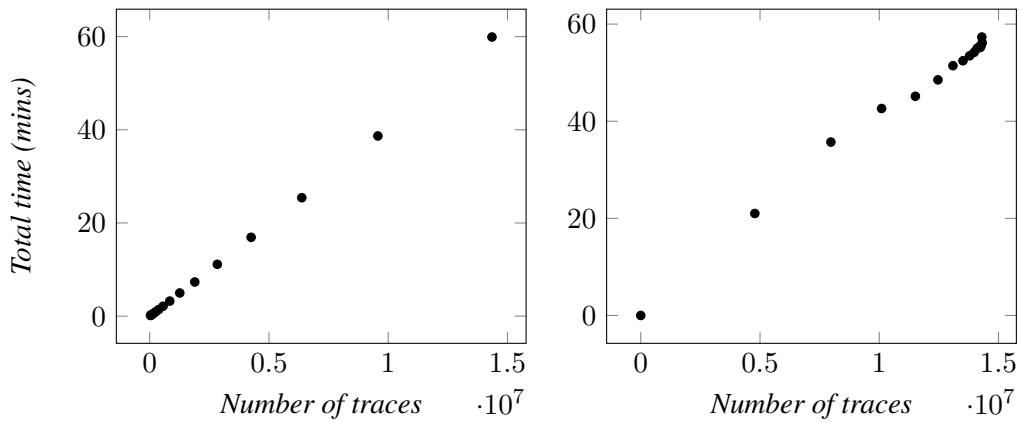
Table 4.1: Summary of results when running the implementation of PEC+ on the tuberculosis scenario. "T Time" column refers to the total computation time in seconds, while "S Time" column refers to solving time in seconds only (the difference between these two values is preprocessing+grounding time). Notice that starting from instant 3, on query $[Tuberculosis = Latent]@I$ (resp. $[Tuberculosis = Active]@I$), the number of traces decreases (resp. increases) when I increases as a result of reactivation.

Query:	$[F=\top]@I$ **holds-with-prob** P				$[F=\bot]@I$ **holds-with-prob** P			
I	P	n Traces	T Time	S Time	P	n Traces	T Time	S Time
0	1	14348907	3594.46	3594.45	0	0	0.009	0
1	0.9	9565938	2320.558	2320.55	0.1	4782969	1260.238	1260.23
2	0.81	6377292	1525.199	1525.19	0.19	7971615	2142.807	2142.8
3	0.729	4251528	1015.366	1015.36	0.271	10097379	2557.311	2557.3
4	0.6561	2834352	667.55	667.54	0.3439	11514555	2708.37	2708.36
5	0.59049	1889568	440.066	440.06	0.40951	12459339	2911.292	2911.28
6	0.531441	1259712	298.561	298.55	0.468559	13089195	3086.318	3086.31
7	0.478297	839808	194.302	194.29	0.521703	13509099	3147.04	3147.03
8	0.430467	559872	127.822	127.81	0.569533	13789035	3208.477	3208.47
9	0.38742	373248	84.424	84.42	0.61258	13975659	3251.099	3251.09
10	0.348678	248832	55.893	55.89	0.651322	14100075	3298.81	3298.8
11	0.313811	165888	37.206	37.2	0.686189	14183019	3316.877	3316.87
12	0.28243	110592	25.531	25.52	0.71757	14238315	3311.27	3311.26
13	0.254187	73728	18.965	18.96	0.745813	14275179	3338.228	3338.22
14	0.228768	49152	12.658	12.65	0.771232	14299755	3440.679	3440.67
15	0.205891	32768	8.88	8.86	0.794109	14316139	3368.671	3368.66

(a) Summary of results when running the implementation of PEC+ on the mock domain description described in example 4.5. "T Time" column refers to the total computation time in seconds, "S Time" column refers to solving time in seconds only (the difference between these two values is preprocessing+grounding time).

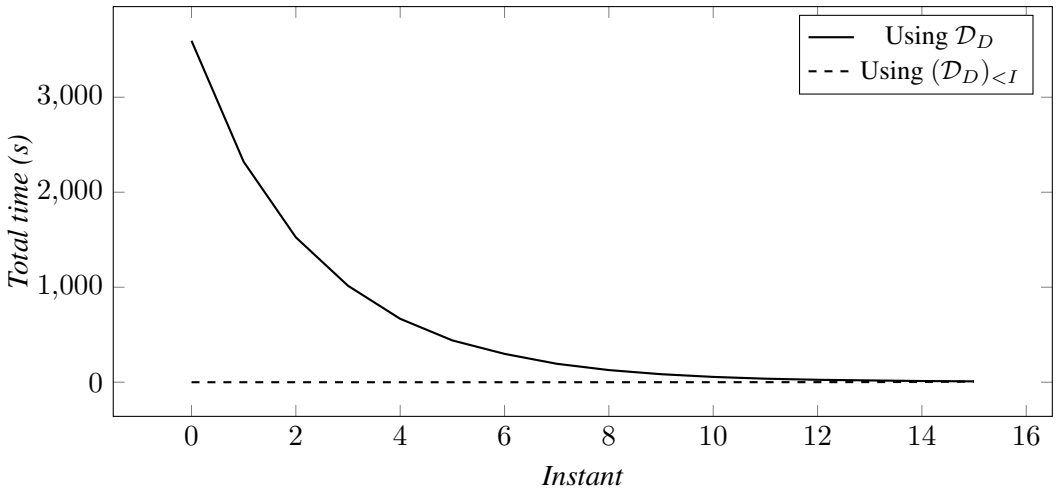


(b) Probability of $F=\top$ and $F=\bot$ at different instants.

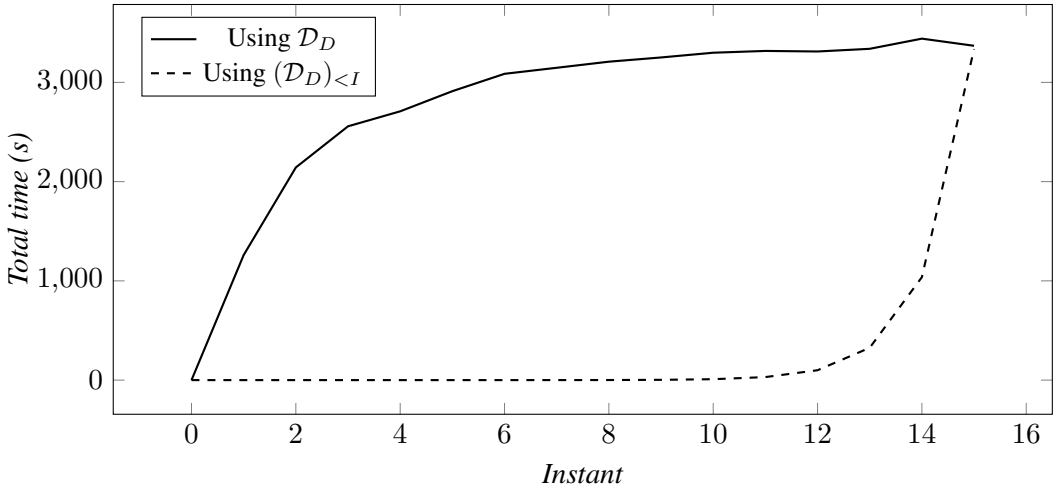


(c) Scatter plots showing the relation between number of traces and total execution time. The plot on the left refers to the computation of answers to the query "$[F=\top]@I$" while the plot on the left refers to the query "$[F=\bot]@I$" for $I \in \{0, 1, \ldots, 15\}$. Correlation is ≈ 0.9997 for the plot on the left and ≈ 0.9962 for the plot on the right.

Figure 4.5



(a) Total execution time on query "$[F = \top]@I$" on (translated) domains $\mathcal{D}_D$ and $(\mathcal{D}_D)_{<I}$.



(b) Total execution time on query "$[F = \bot]@I$" on (translated) domains $\mathcal{D}_D$ and $(\mathcal{D}_D)_{<I}$.

Figure 4.6

Chapter 5

Language EPEC

This chapter introduces EPEC (short for *Epistemic Probabilistic Event Calculus*), a language that goes beyond PEC+ (introduced in chapter 3 and implemented in ASP in chapter 4) in the sense that it allows the agent to *sense* the environment through actions that produce *knowledge* about the value of fluents rather than an effect on the environment. Furthermore, it allows for the definition of actions that are conditioned on knowledge produced by sensing actions.

5.1 Syntax

The definition of Domain Language in EPEC is very similar to the definition of Domain Language in PEC+. A slight difference is that the function *vals* now maps fluents and actions to *tuples* of elements. This is to induce a "standard" ordering over values a fluent can take, which is then exploited in the subsequent definitions.

Definition 5.1 (Domain Language)**.** A *domain language* for EPEC is a tuple $\langle \mathcal{F}, \mathcal{A}, \mathcal{V}, \mathit{vals}, \mathcal{I}, \leq, \bar{0} \rangle$ consisting of a finite non-empty set $\mathcal{F}$ of *fluents*, a finite set $\mathcal{A}$ of *actions*, a finite non-empty set $\mathcal{V}$ of *values* such that $\{\top, \bot\} \subseteq \mathcal{V}$, a function *vals* mapping elements in $\mathcal{F} \cup \mathcal{A}$ to tuples of elements (without repetitions) from $\mathcal{V}$, a non-empty set $\mathcal{I}$ of *instants* and a minimum element $\bar{0} \in \mathcal{I}$ w.r.t. a total ordering $\leq$ over $\mathcal{I}$. For $A \in \mathcal{A}$ it is imposed that $\mathit{vals}(A) = \langle \top, \bot \rangle$ and for any $X \in \mathcal{F} \cup \mathcal{A}$ the expression $V \in \mathit{vals}(X)$ means that if $\mathit{vals}(X) = \langle V_1, \ldots, V_n \rangle$ then $V = V_i$ for some $1 \leq i \leq n$.

Example 5.1. An appropriate domain language for scenario 1.6 is

$$\langle \mathcal{F}_L, \mathcal{A}_L, \mathcal{V}_L, \mathit{vals}, \mathbb{N}, \leq_{\mathbb{N}}, 0 \rangle$$

where

$$\begin{gathered} \mathcal{F}_L = \{\mathit{Light}\}, \\ \mathcal{A}_L = \{\mathit{Press}, \mathit{SenseLight}\}, \\ \mathcal{V}_L = \{\top, \bot, \mathit{On}, \mathit{Off}\}, \\ \mathit{vals}_L(\mathit{Light}) = \langle \mathit{On}, \mathit{Off} \rangle \end{gathered}$$

where, as usual, $\mathbb{N}$ is the set of natural numbers (including 0), and $\leq_{\mathbb{N}}$ is the standard total ordering between naturals.

In what follows, all definitions are with respect to a domain language $\langle \mathcal{F}, \mathcal{A}, \mathcal{V}, \mathit{vals}, \mathcal{I}, \leq, \bar{0} \rangle$. Note that the syntax of EPEC borrows some elements from PEC+. For instance, the definitions of *i-proposition*, *c-proposition*, *formula*, *i-formula* and *(fluent) states* are as in PEC+.

Definition 5.2 (s-proposition, Body of an s-proposition)**.** Let $X \in \mathcal{F} \cup \mathcal{A}$ and $\mathit{vals}(X) = \langle V_1, \ldots, V_m \rangle$. An *s-proposition* s has the form

$$\theta \textbf{ senses } X \textbf{ with-accuracies } \mathbf{M} \tag{5.1}$$

where θ entails some $A \in \mathcal{A}$, and $\mathbf{M}$ is an $m \times m$ matrix with elements in $[0, 1]$. For an s-proposition s of the generic form (5.1), θ is called the *body of* s and denoted by $\mathit{body}(s)$, and X is called the *object of* s. The pair (θ, X) is called the *signature of* s and denoted by $\mathit{sign}(s)$.

Intuitively, every element $\mathbf{M}_{i,j}$ in $\mathbf{M}$ represents the probability that, given that V_i is the real value taken by X when the s-proposition occurs, V_j is sensed instead.

$\mathbf{M}$ is subject to the condition[1]:

$$\forall 1 \leq i \leq m, \quad \sum_{j=1}^{m} \mathbf{M}_{i,j} = 1 \tag{5.2}$$

or, in matricial form,

$$\mathbf{M} \times \mathbf{1}_m = \mathbf{1}_m$$

i.e. $\mathbf{1}_m$ is a right eigenvector of $\mathbf{M}$ with eigenvalue 1, where $\mathbf{1}_m$ is the $m \times 1$ vector $\begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$

The s-proposition

$$\theta \textbf{ senses } X \textbf{ with-accuracies } \mathbf{I}_M$$

where $\mathbf{I}_M$ is the $m \times m$ identity matrix, is sometimes abbreviated to

$$\theta \textbf{ senses } X$$

Definition 5.3 (Epistemic p-proposition)**.** An *(epistemic) p-proposition* p has the following form:

$$A \textbf{ performed-at } I \textbf{ with-prob } P^+ \textbf{ if-believes } (\theta, \bar{P}) \tag{5.3}$$

for some action A, instant I, $P^+ \in (0, 1]$, formula θ, and some (open, half-open or closed) interval $\bar{P}$ with endpoints in $[0, 1]$. The pair $(\theta, \bar{P})$ is called the *body of* p and denoted by $\mathit{body}(p)$. When an (epistemic) p-proposition p has the form (5.3), then it is said that p *has instant* I.

Definition 5.4 (o-proposition)**.** An *o-proposition* o has the form

$$A \textbf{ occurs-at } I \textbf{ with-prob } P^+ \textbf{ if-holds } \theta$$

for some action A, instant I, $P^+ \in (0, 1]$ and formula θ. The formula θ is called the *body of* o

[1] A matrix satisfying this condition is sometimes said to be a *right stochastic matrix*.

and denoted by $body(o)$. When an o-proposition o has this form, then it is said that o *has instant* I.

Definition 5.5 (Domain Description)**.** A *domain description* is a finite set $\mathcal{D}$ of v-propositions, c-propositions, p-propositions, o-propositions, i-propositions and s-propositions such that: (i) for any two distinct c-propositions in $\mathcal{D}$ with bodies θ and η, θ is incompatible with η (i.e., there is no state S such that $S \models \theta$ and $S \models \eta$), (ii) $\mathcal{D}$ contains exactly one i-proposition, (iii) $\mathcal{D}$ contains exactly one v-proposition for each $F \in \mathcal{F}$ and (iv) if an o-proposition "A **occurs-at** I **with-prob** P^+ **if-holds** θ" belongs to $\mathcal{D}$, then for all $P' \in (0, 1]$ and formulas η that are compatible with θ (i.e. such that for some state S both $S \models \theta$ and $S \models \eta$ hold) there is no other o-proposition of the form "A **occurs-at** I **with-prob** P' **if-holds** η" that belongs to $\mathcal{D}$, (v) no two s-propositions in $\mathcal{D}$ have the same signature.

Notation 5.1. For a fixed domain description $\mathcal{D}$ and each pair (θ, X) there exists at most one s-proposition s in $\mathcal{D}$ such that $sign(s) = (\theta, X)$, therefore $\mathbf{M}_{\mathcal{D}}(\theta, X)$ can be taken to denote the matrix $\mathbf{M}$ such that the s-proposition "θ **senses** X **with-accuracies** $\mathbf{M}$" is in $\mathcal{D}$. The subscript $\mathcal{D}$ is sometimes dropped when the reference to the domain description is clear from the context.

The following definition defines an (often desirable) syntactical property of domains. Intuitively, a domain description satisfying it is meant to represent an agent that is aware of the actions being performed.

Definition 5.6 (Awareness Property)**.** An EPEC domain description $\mathcal{D}$ is said to satisfy the *awareness property* if for each epistemic p-proposition

$$A \textbf{ performed-at } I \textbf{ with-prob } P^+ \textbf{ if-believes } (\theta, \bar{P})$$

in $\mathcal{D}$ there exists in $\mathcal{D}$ an s-proposition of the form

$$A \textbf{ senses } A$$

If a domain description does not satisfy the awareness property, axioms can be added such that the augmented domain description satisfies it. The macro

awareness-property-on

is a shorthand for the set of axioms that make the domain description satisfy the awareness property.

Example 5.2. A suitable domain description $\mathcal{D}_L$ for Scenario 1.6 is the following:

(L1) *Light* **takes-values** $\langle On, Off \rangle$

(L2) **initially-one-of** $\{(\{Light = On\}, 0.5), (\{Light = Off\}, 0.5)\}$

(L3) $Press \wedge Light = On$ **causes-one-of** $\{(\{Light = Off\}, 0.8)\}$

(L4) $Press \wedge Light = Off$ **causes-one-of** $\{(\{Light = On\}, 0.9)\}$

(L5) *SenseLight* **senses** *Light* **with-accuracies** $\begin{pmatrix} 0.95 & 0.05 \\ 0.15 & 0.85 \end{pmatrix}$

(L6) *SenseLight* **performed-at** 0

(L7) *Press* **performed-at** 1 **if-believes** $(\mathit{Light} = \mathit{Off}, [0.9, 1])$

For the remainder of this chapter, $\mathcal{D}$ is an arbitrary domain description.

5.2 Semantics

The semantics of EPEC relies again on the concept of a *world*, which is exactly the same as in PEC+. In addition, EPEC's semantics makes use of *sensing outcomes* and *histories*, which are intended to record the result of sensing actions performed by the agent. Pairs constiting of a world and a sensing history are called *h-worlds*.

Definition 5.7 (Worlds)**.** A *world* is a function $W : \mathcal{I} \to \mathcal{S}$. The set of all worlds is denoted by $\mathcal{W}$.

Definition 5.8 (Sensing outcome)**.** A *sensing outcome* is a pair of the form $((\theta, X), V)$ for some signature (θ, X) and a value $V \in \mathit{vals}(X)$.

Definition 5.9 (Sensing history, Indistinguishable sensing histories)**.** A *sensing history* is a function H from I to the power set of sensing outcomes. The set of all sensing histories is denoted by $\mathcal{H}$. Two sensing histories H and H' are *indistinguishable up to (and excluding) I* if and only if $H(I') = H'(I')$ for all $I' < I$. The set consisting of all indistinguishable sensing histories H' and H up to I forms an equivalence class and is denoted by $[H]_{<I}$. The class $[H]_{<I}$ is sometimes represented in the form

$$\langle H(I_1)@I_1, \ldots, H(I_n)@I_n \rangle$$

for instants $I_1, \ldots, I_n$ such that $I_i < I$ and $H(I_i) \neq \emptyset$ for $1 \leq i \leq n$ and notice that for any $H \in \mathcal{H}$, $\langle\rangle$ stands for $[H]_{<\bar{0}}$.

Notation 5.2. In the following, $H, H', H_1, H'', H_2, \ldots$ will denote sensing histories.

Definition 5.10 (h-world)**.** An *h-world* is a pair (W, H) for a world $W \in \mathcal{W}$ and a sensing history $H \in \mathcal{H}$.

Example 5.3. Consider the following worlds:

$W_1(0) = \{\mathit{Light} = \mathit{On}, \neg \mathit{Press}, \mathit{SenseLight}\}$
$W_1(I) = \{\mathit{Light} = \mathit{On}, \neg \mathit{Press}, \neg \mathit{SenseLight}\}$ for $I > 0$

$W_2(0) = \{\mathit{Light} = \mathit{Off}, \neg \mathit{Press}, \mathit{SenseLight}\}$
$W_2(I) = \{\mathit{Light} = \mathit{Off}, \neg \mathit{Press}, \neg \mathit{SenseLight}\}$ for $I > 0$

$W_3(0) = \{\mathit{Light} = \mathit{On}, \neg \mathit{Press}, \mathit{SenseLight}\}$
$W_3(1) = \{\mathit{Light} = \mathit{On}, \mathit{Press}, \neg \mathit{SenseLight}\}$

$W_3(I) = \{Light = Off, \neg Press, \neg SenseLight\}$ for $I > 1$

$W_4(0) = \{Light = Off, \neg Press, SenseLight\}$
$W_4(1) = \{Light = Off, Press, \neg SenseLight\}$
$W_4(I) = \{Light = On, \neg Press, \neg SenseLight\}$ for $I > 1$

$W_5(0) = \{Light = On, \neg Press, SenseLight\}$
$W_5(1) = \{Light = On, Press, \neg SenseLight\}$
$W_5(I) = \{Light = On, \neg Press, \neg SenseLight\}$ for $I > 1$

$W_6(0) = \{Light = Off, \neg Press, SenseLight\}$
$W_6(1) = \{Light = Off, Press, \neg SenseLight\}$
$W_6(I) = \{Light = Off, \neg Press, \neg SenseLight\}$ for $I > 1$

$W_7(0) = \{Light = On, \neg Press, \neg SenseLight\}$
$W_7(1) = \{Light = On, Press, SenseLight\}$
$W_7(I) = \{Light = Off, \neg Press, \neg SenseLight\}$ for $I > 1$

and the following sensing histories:

$H_1(0) = \{(sign(s), Light = On)\}$
$H_1(I) = \emptyset$ for $I > 0$

$H_2(0) = \{(sign(s), Light = Off)\}$
$H_2(I) = \emptyset$ for $I > 0$

$H_3(I) = \{(sign(s), Light = On)\}$ for $I \leq 1$
$H_3(I) = \emptyset$ for $I > 1$

$H_4(0) = \{(sign(s), Light = On)\}$
$H_4(1) = \{(sign(s), Light = Off)\}$
$H_4(I) = \emptyset$ for $I > 1$

where s is the only s-proposition in $\mathcal{D}_L$, i.e. (L5). Then, each (W_i, H_j) for $i \in \{1, 2, 3, 6, 7\}$ and $j \in \{1, 2, 3, 4\}$ is an h-world for Scenario 1.6.

The following definition is similar to definition 3.17 in PEC+:

Definition 5.11 (Sensing Occurrence)**.** Let $\mathcal{D}$ be a domain description, θ be the body of an s-proposition s in $\mathcal{D}$ and $I \in \mathcal{I}$. If $W \mid\!\models [\theta]@I$ then it is said that that *a sensing action occurs at instant I in W w.r.t. to $\mathcal{D}$*, and that *the s-proposition s is activated at I in W w.r.t. $\mathcal{D}$*. Let (W, H) be an h-world. For any instant $I \in \mathcal{I}$, the set $socc_{\mathcal{D}}((W, H), I)$ is defined as $\{((\theta, X), V, V') \mid sign(s) = (\theta, X), W \mid\!\models [\theta \wedge X = V]@I, ((\theta, X), V') \in H(I)\}$ if there exists at least one sensing proposition s that is activated in W at I w.r.t. $\mathcal{D}$, and is empty otherwise.

The following definition plays a role similar to definition 3.16 in PEC+. H-worlds satisfying it are those in which the sensing history is producing a result which is consistent with what the agent decided to sense through the performance of a sensing action.

Definition 5.12 (CWA for Sensing Actions)**.** An h-world (W, H) satisfies the *closed world assumption for sensing actions w.r.t.* $\mathcal{D}$ (*CWAS* for short) if and only if for each s-proposition s activated at I in W, $(\mathit{sign}(s), V) \in H(I)$ for some $V \in \mathit{vals}(F)$, and $\forall V' \neq V$, $(\mathit{sign}(s), V') \notin H(I)$.

Given a domain description $\mathcal{D}$ and world W, the set of histories

$$\{H \in \mathcal{H} \mid (W, H) \text{ satisfies CWAS w.r.t. } \mathcal{D}\}$$

is denoted by $\mathcal{H}^W_{\mathcal{D}}$.

Example 5.4. Recall the domain description $\mathcal{D}_L$ from example 5.2 and the h-worlds from Example 5.3. H-worlds (W_7, H_1), ..., (W_7, H_4) do not satisfy CWAS w.r.t. $\mathcal{D}_L$ as only one sensing action is being activated at 1 in W_7, but $H_1(0)$, $H_2(0)$, $H_3(0)$ and $H_4(0)$ are all non-empty. On the other hand, each (W_i, H_j) for $i \in \{1, 2, 3, 4, 5, 6\}$ and $j \in 1, 2$ satisfies CWAS w.r.t. $\mathcal{D}_L$.

Definition 5.13 (Evaluation of a sensing history)**.** Let $\mathcal{D}$ be a domain-description and (W, H) be an h-world. The *evaluation of* H *given* W *w.r.t.* $\mathcal{D}$ is defined as

$$\epsilon_{\mathcal{D}}(H \mid W) = \begin{cases} \prod_{I \in \mathcal{I}} \left(\prod_{((\theta, X), V_i, V_j) \in \mathit{socc}_{\mathcal{D}}((W,H),I)} \mathbf{M}(\theta, X)_{i,j} \right) & \text{if } (W, H) \text{ sat. CWAS} \\ 0 & \text{otherwise} \end{cases}$$

and notice that satisfaction of CWAS w.r.t. $\mathcal{D}$ implies that $\mathit{socc}((W, H), I)$ is non empty for only finitely many instants, hence the above product is always well-defined. For a class $[H]_{<I}$ of indistinguishable sensing histories up to I, the sum

$$\sum_{H' \in [H]_{<I}} \epsilon_{\mathcal{D}}(H' \mid W)$$

is denoted by $\epsilon_{\mathcal{D}}([H]_{<I} \mid W)$.

Example 5.5. As shown in example 5.4, h-worlds (W_1, H_1), (W_1, H_2), (W_2, H_1), (W_2, H_2), (W_3, H_1), (W_3, H_2), (W_4, H_1), (W_4, H_2) satisfy CWAS w.r.t. $\mathcal{D}_L$. Then,

$$\epsilon_{\mathcal{D}_L}(H_1 \mid W_1) = 0.95 \quad \epsilon_{\mathcal{D}_L}(H_1 \mid W_2) = 0.15 \quad \epsilon_{\mathcal{D}_L}(H_1 \mid W_3) = 0.95 \quad \epsilon_{\mathcal{D}_L}(H_1 \mid W_4) = 0.15$$
$$\epsilon_{\mathcal{D}_L}(H_2 \mid W_1) = 0.05 \quad \epsilon_{\mathcal{D}_L}(H_2 \mid W_2) = 0.85 \quad \epsilon_{\mathcal{D}_L}(H_2 \mid W_3) = 0.05 \quad \epsilon_{\mathcal{D}_L}(H_2 \mid W_4) = 0.85$$

Since h-worlds (W_7, H_1), ..., (W_7, H_4) do not satisfy CWAS, they evaluate to 0.

In order to define what a well-behaved h-world is in this setting, some auxiliary definitions and machinery is needed.

Recall that using the product rule (2.6), a joint probability distribution d over two random variables X, Y can be written e.g. as $d(X, Y) = d(Y \mid X)d(X)$ for some *marginal probability*

$d(X)$ and some *conditional probability* $d(Y \mid X)$. Therefore it is possible to construct a joint probability distribution over worlds and histories by simply defining the marginal $d(W)$ and the conditional $d(H \mid W)$. This is the idea behind the following definition:

Definition 5.14 (Pre-model)**.** Given an EPEC domain description $\mathcal{D}$ and a PEC+ domain description $\mathcal{D}'$, the *pre-model* $\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}$ *of* $\mathcal{D}$ *w.r.t.* $\mathcal{D}'$ is a [0,1]-interpretation over $\mathcal{W} \times \mathcal{H}$ defined as:

$$\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}(W, H) = M_{\mathcal{D}'}(W) \cdot e_{\mathcal{D}}(H \mid W).$$

As usual, $\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}(W, [H]_{<I})$ is shorthand for

$$\sum_{H \in [H]_{<I}} \tilde{M}_{\mathcal{D}}^{\mathcal{D}'}(W, H)$$

and $\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}(\varphi, H)$ is shorthand for

$$\sum_{W \mid\!= \varphi} \tilde{M}_{\mathcal{D}}^{\mathcal{D}'}(W, H)$$

Informally, $\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}(W, H)$ defines a joint probability distribution over $\mathcal{W} \times \mathcal{H}$ by taking $M_{\mathcal{D}'}(W)$ as the marginal distribution and $e_{\mathcal{D}}(H \mid W)$ as the conditional distribution. The following proposition is the first step towards showing that $\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}(W, H)$ is indeed a probability function, and shows that the evaluation of a class of indistinguishable sensing histories only depends on instants up to I' (and compare this with definition 5.13):

Proposition 5.1. Let $\mathcal{D}$ be an EPEC domain description and (W, H) be a world satisfying CWAS w.r.t. $\mathcal{D}$. Then,

$$\epsilon_{\mathcal{D}}([H]_{<I} \mid W) = \prod_{I' < I} \left(\prod_{((\theta,X),V_i,V_j) \in \mathit{socc}_{\mathcal{D}}((W,H),I')} \mathbf{M}(\theta, X)_{i,j} \right)$$

Proof. By definition ,

$$\epsilon_{\mathcal{D}}([H]_{<I} \mid W) = \sum_{H' \in [H]_{<I}} \epsilon_{\mathcal{D}}(H' \mid W) = \sum_{H' \in [H]_{<I}} \left(\prod_{I \in \mathcal{I}} \left(\prod_{((\theta,X),V_i,V_j) \in \mathit{socc}_{\mathcal{D}}((W,H'),I)} \mathbf{M}(\theta, X)_{i,j} \right) \right)$$

Recall that $[H]_{<I}$ is an equivalence class (meaning that if $H', H'' \in [H]_{<I}$ then $H'(I') = H''(I')$ for all $I' < I$). Then, for any $X \in \mathcal{F} \cup \mathcal{A}$, $V_i \in \mathit{vals}(X)$ and $I' \geq I$, if $((\theta, X), V_i) \in H(I')$ then for all $V_j \neq V_i$ such that $V_j \in \mathit{vals}(X)$ there exists some other $H' \in [H]_{<I}$ such that (W, H') also satisfies CWAS w.r.t. $\mathcal{D}$ and $((\theta, X), V_j) \in H'(I')$. Therefore the above sum can be rewritten as:

$$= \underbrace{\prod_{I' < I} \left(\prod_{((\theta,X),V_i,V_j) \in \mathit{socc}_{\mathcal{D}}((W,H),I')} \mathbf{M}(\theta, X)_{i,j} \right)}_{\text{common factor to all } H \in [H]_{<I}} \prod_{I' \geq I} \left(\sum_{((\theta,X),V_i) \in H(I')} \left(\sum_{j=1}^{m} \mathbf{M}(\theta, X)_{i,j} \right) \right)$$

and since by eq. (5.2) $\sum_{j=1}^{m} \mathbf{M}(\theta, X)_{i,j} = 1$ the proposition is proved. □

Corollary 5.1.1. Let $\mathcal{D}$ be an EPEC domain description and W a world. Then,

$$\sum_{H \in \mathcal{H}} \epsilon_{\mathcal{D}}(H \mid W) = 1.$$

Proof. Follows directly from proposition 5.1 by considering the equivalence class $[H]_{<\bar{0}} = \mathcal{H}$. □

Corollary 5.1.2. Let $\mathcal{D}$ be an EPEC domain description and $\mathcal{D}'$ a PEC+ domain description. The *pre-model* $\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}$ *of* $\mathcal{D}$ *w.r.t.* $\mathcal{D}'$ is a probability distribution in the sense that it satisfies eq. (2.4) from section 2.2.1.

Proof. Follows directly from the product rule (2.6) and the definition of $\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}$ since $M_{\mathcal{D}'}$ is a probability distribution and for each $W \in \mathcal{W}$ also $\epsilon_{\mathcal{D}}(\cdot \mid W)$ is a probability distribution (see corollary 5.1.1). □

Given that $\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}$ is a probability distribution, it is possible to use the sum rule 2.5, product rule 2.6 and Bayes' theorem 2.8 to derive some other quantities such as $\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}(W \mid H)$, which is particularly important as it represents the probability of W being the actual world from the point of view of an agent that has sensing history H.

Definition 5.15 (Reduct of a p-proposition)**.** Let "A **performed-at** I **with-prob** P^+ **if-believes** $(\theta, \bar{P})$" be an epistemic p-proposition and d some probability distribution (over worlds). Then, this p-proposition *reduces to a (non-epistemic) p-proposition "*A **performed-at** I **with-prob** P^+*" w.r.t.* d if and only if $\sum_{W \mid\models [\theta]@I} d(W) \in \bar{P}$, and vice-versa "$A$ **performed-at** I **with-prob** P^+" is said to be the *reduct of "*A **performed-at** I **with-prob** P^+ **if-believes** $(\theta, \bar{P})$*" w.r.t.* d.

Example 5.6. Consider the (only) epistemic p-proposition *L7* in $\mathcal{D}_L$, i.e.

$$\textit{Press} \textbf{ performed-at } 1 \textbf{ if-believes } (\textit{Light} = \textit{On}, [0.9, 1]),$$

let W_1 and W_2 be as in example 5.3 and d be a distribution such that $d(W_1) = 1$ and $d(W) = 0$ for all $W \neq W_1$. Then, the reduct of *L7* w.r.t. d is

$$\textit{Press} \textbf{ performed-at } 1.$$

If instead d is the distribution such that $d(W_2) = 1$ and $d(W_2) = 0$ for all $W \neq W_2$, then *L7* has no reduct w.r.t. d.

Definition 5.16 (Reduct of a Domain Description)**.** Let $\mathcal{D}$ be an EPEC domain description, d be a probability distribution (over worlds and histories) and H a sensing history. Then, *the reduct of* $\mathcal{D}$ *w.r.t.* d *and* H is the PEC+ domain description obtained by removing all the s-propositions, o-propositions and epistemic p-propositions from $\mathcal{D}$ and (i) including one p-proposition of the form "A **performed-at** I **with-prob** P^+ **if-holds** θ" whenever "A **occurs-at** I **with-prob** P^+ **if-holds** θ" belongs to $\mathcal{D}$, and (ii) including the reducts of each epistemic p-proposition "A **performed-at** I **with-prob** P^+ **if-believes** $(\theta, \bar{P})$" in $\mathcal{D}$ w.r.t. the probability distribution $d(\cdot \mid [H]_{<I})$.

Definition 5.17 (Candidate Reduct of a Domain Description)**.** Let $\mathcal{D}$ be an EPEC domain description. If there exists a distribution d such that $\mathcal{D}'$ is a reduct of $\mathcal{D}$ w.r.t. d, then $\mathcal{D}'$ is said to be a *candidate reduct of* $\mathcal{D}$.

Example 5.7. Since there is only one epistemic p-proposition in $\mathcal{D}_L$, there are only two candidate reducts of $\mathcal{D}_L$, namely the PEC+ domain description $\mathcal{D}'_L$:

Light **takes-values** $\{On, Off\}$
initially-one-of $\{(\{Light = On\}, 0.5), (\{Light = Off\}, 0.5)\}$
Press $\wedge$ *Light* $=$ *On* **causes-one-of** $\{(\{Light = Off\}, 0.9)\}$
Press $\wedge$ *Light* $=$ *Off* **causes-one-of** $\{(\{Light = On\}, 0.8)\}$
SenseLight **performed-at** 0

and the PEC+ domain description $\mathcal{D}''_L$:

Light **takes-values** $\{On, Off\}$
initially-one-of $\{(\{Light = On\}, 0.5), (\{Light = Off\}, 0.5)\}$
Press $\wedge$ *Light* $=$ *On* **causes-one-of** $\{(\{Light = Off\}, 0.9)\}$
Press $\wedge$ *Light* $=$ *Off* **causes-one-of** $\{(\{Light = On\}, 0.8)\}$
SenseLight **performed-at** 0
Press **performed-at** 1

For any EPEC domain description $\mathcal{D}$ containing exactly n epistemic p-propositions, there are at least 1 and at most 2^n PEC+ domain descriptions that are candidate reducts of $\mathcal{D}$.

Definition 5.18. Let $\mathcal{D}$ be an EPEC domain description and H a sensing history. Then, the set $R(\mathcal{D}, H)$ is the set of PEC+ domain descriptions such that $\mathcal{D}' \in R(\mathcal{D}, H)$ if and only if (i) $\mathcal{D}'$ is a candidate reduct of $\mathcal{D}$, (ii) $\mathcal{D}'$ is the reduct of $\mathcal{D}$ w.r.t. the pre-model $\tilde{M}^{\mathcal{D}'}_{\mathcal{D}}$ and history H.

Example 5.8. There are two candidate reducts of $\mathcal{D}_L$, hence one must consider the two pre-models $\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}_L}$ and $\tilde{M}^{\mathcal{D}''_L}_{\mathcal{D}_L}$ where $\mathcal{D}'_L$ and $\mathcal{D}''_L$ are defined as in example 5.7. There are two well-behaved worlds w.r.t. $\mathcal{D}'_L$ (namely, W_1 and W_2 as in example 5.3) and four well-behaved worlds w.r.t. $\mathcal{D}''_L$ (namely, $W_3 \ldots W_6$ as in example 5.3). Consider $\mathcal{D}'_L$. This is such that:

$$M_{\mathcal{D}'_L}(W_1) = M_{\mathcal{D}'_L}(W_2) = 0.5$$

and now consider the two h-worlds (W_1, H_1) and (W_2, H_1) satisfying CWAS. Evaluating $\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}_L}(\cdot \mid [H_1]_{<I})$ gives:

$$\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}_L}(W_1 \mid [H_1]_{<1}) = \frac{0.5 \cdot 0.95}{0.5 \cdot 0.95 + 0.5 \cdot 0.15} \approx 0.864$$

and

$$\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}'_L}(W_2 \mid [H_1]_{<1}) = \frac{0.5 \cdot 0.15}{0.5 \cdot 0.95 + 0.5 \cdot 0.15} \approx 0.136$$

and therefore $\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}'_L}([\mathit{Light} = \mathit{Off}]@1 \mid [H_1]_{<1}) \approx 0.136 \notin [0.9, 1]$. Calculating the reduct of $\mathcal{D}_L$ w.r.t. $\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}'_L}$ then gives $\mathcal{D}'_L$ itself. Hence $\mathcal{D}'_L \in R(\mathcal{D}_L, H_1)$.

Now consider the two worlds (W_1, H_2) and (W_2, H_2) also satisfying CWAS. Evaluating $\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}'_L}(\cdot \mid [H_2]_{<I})$ gives:

$$\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}'_L}(W_1 \mid [H_2]_{<1}) = \frac{0.5 \cdot 0.05}{0.5 \cdot 0.05 + 0.5 \cdot 0.85} \approx 0.056$$

and

$$\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}'_L}(W_2 \mid [H_2]_{<1}) = \frac{0.5 \cdot 0.05}{0.5 \cdot 0.95 + 0.5 \cdot 0.85} \approx 0.944$$

and therefore $\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}'_L}([\mathit{Light} = \mathit{Off}]@1 \mid [H_2]_{<2}) \approx 0.944 \in [0.9, 1]$. Calculating the reduct of $\mathcal{D}_L$ w.r.t. $\tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}'_L}$ then gives $\mathcal{D}''_L$. Hence $\mathcal{D}'_L \notin R(\mathcal{D}_L, H_2)$.

A similar reasoning leads to the conclusion that $\mathcal{D}''_L \notin R(\mathcal{D}, H_1)$ and $\mathcal{D}''_L \in R(\mathcal{D}, H_2)$, where

$$\tilde{M}^{\mathcal{D}''_L}_{\mathcal{D}'_L}(W_3 \mid [H_2]_{<1}) \approx 0.044 \qquad \tilde{M}^{\mathcal{D}''_L}_{\mathcal{D}'_L}(W_4 \mid [H_2]_{<1}) \approx 0.85$$
$$\tilde{M}^{\mathcal{D}''_L}_{\mathcal{D}'_L}(W_5 \mid [H_2]_{<1}) \approx 0.011 \qquad \tilde{M}^{\mathcal{D}''_L}_{\mathcal{D}'_L}(W_6 \mid [H_2]_{<1}) \approx 0.094$$

Proposition 5.2. Given an EPEC domain description $\mathcal{D}$ and a sensing history H, $R(\mathcal{D}, H)$ is in fact a singleton set.

Proof. By contradiction. Let $\mathcal{D}'$ and $\mathcal{D}''$ be two distinct PEC+ domain descriptions in $R(\mathcal{D}, H)$. Since they are different, one can consider the set of p-propositions for which they differ $\{p \in \mathit{narr}(\mathcal{D}') \cup \mathit{narr}(\mathcal{D}'') \mid (p \in \mathcal{D}' \wedge p \notin \mathcal{D}'') \vee (p \notin \mathcal{D}' \wedge p \in \mathcal{D}'')\}$. Consider an ordering of these p-propositions by their instants and let p be the first in such ordering. Without loss of generality, let p have instant I and $p \in \mathcal{D}'$ but $p \notin \mathcal{D}''$. By definition, p must be the reduct of some epistemic p-proposition in $\mathcal{D}$ and let its body be $(\theta, \bar{P})$. Since $p \in \mathcal{D}'$ and $p \notin \mathcal{D}''$, $\tilde{M}^{\mathcal{D}'}_{\mathcal{D}}(\theta \mid [H]_{<I}) \in \bar{P}$ and $\tilde{M}^{\mathcal{D}''}_{\mathcal{D}}(\theta \mid [H]_{<I}) \notin \bar{P}$ implying

$$\frac{\tilde{M}^{\mathcal{D}'}_{\mathcal{D}}([\theta]@I, [H]_{<I})}{\tilde{M}^{\mathcal{D}'}_{\mathcal{D}}([H]_{<I})} \neq \frac{\tilde{M}^{\mathcal{D}''}_{\mathcal{D}}([\theta]@I, [H]_{<I})}{\tilde{M}^{\mathcal{D}''}_{\mathcal{D}}([H]_{<I})}$$

To prove that this is not the case, it suffices to show that $\tilde{M}^{\mathcal{D}'}_{\mathcal{D}}([\theta]@I, [H]_{<I}) = \tilde{M}^{\mathcal{D}''}_{\mathcal{D}}([\theta]@I, [H]_{<I})$ for an arbitrary θ, as this also accounts for $\tilde{M}^{\mathcal{D}'}_{\mathcal{D}}([H]_{<I}) = \tilde{M}^{\mathcal{D}''}_{\mathcal{D}}([H]_{<I})$ by letting θ be any tautology. Let $\{W_1, \ldots, W_m\}$ be a maximal set of distinct representatives of indistinguishable classes of worlds up to I and let $[W_1]_I, \ldots, [W_m]_I$ be the classes they represent. Also let $\varphi_{[W]_I}$ be an i-formula[2] that "captures" the class of indistinguishable worlds up to I, i.e. $W \models \varphi_{[W]_I}$ and $W' \models \varphi_{[W]_I}$ if and only if W and W' are indistinguishable up to I. Take φ_i as a shorthand for the conjunction $[\theta]@I \wedge \varphi_{[W_i]_I}$ and notice that it only depends on

[2]This i-formula exists as it is possible e.g. to consider an appropriate conjunction of states in W up to I as to match definition 3.37.

instants $< I$. Then,

$$\tilde{M}_{\mathcal{D}}^{\mathcal{D}'}([\theta]@I, [H]_{<I}) = \sum_{W|\models[\theta]@I} \epsilon_{\mathcal{D}}([H]_{<I} \mid W) M_{\mathcal{D}'}(W) = \sum_{i=1}^{m} \left(\sum_{W|\models\varphi_i} \epsilon_{\mathcal{D}}([H]_{<I} \mid W) M_{\mathcal{D}'}(W) \right)$$

and notice that proposition 5.1 implies that $\epsilon_{\mathcal{D}}([H]_{<I} \mid W)$ only depends on values of W up to instant I. Therefore, $\epsilon_{\mathcal{D}}([H]_{<I} \mid W)$ has a constant value whenever $W \mid\models \varphi_i$, and let this constant value be denoted by $\epsilon_{\mathcal{D}}([H]_{<I} \mid \varphi_i)$. The above sum therefore continues as follows:

$$= \sum_{i=1}^{m} \epsilon_{\mathcal{D}}([H]_{<I} \mid \varphi_i) \left(\sum_{W|\models\varphi_i} M_{\mathcal{D}'}(W) \right) = \sum_{i=1}^{m} \epsilon_{\mathcal{D}}([H]_{<I} \mid \varphi_i) M_{\mathcal{D}'}(\varphi_i)$$

But then, since the two domain descriptions $\mathcal{D}'$ and $\mathcal{D}''$ disagree only on p-propositions having instant $\geq I$, $M_{\mathcal{D}'}(\varphi_i) = M_{\mathcal{D}''}(\varphi_i)$ for all $1 \leq i \leq m$ and therefore the above sum continues

$$= \sum_{i=1}^{m} \epsilon_{\mathcal{D}}([H]_{<I} \mid \varphi_i) M_{\mathcal{D}''}(\varphi_i) = \tilde{M}_{\mathcal{D}}^{\mathcal{D}''}(\theta, [H]_{<I})$$

which proves the proposition.

□

Since $R(\mathcal{D}, H)$ is a singleton set, $\mathcal{D}_H$ can be taken to denote the unique domain description in $R(\mathcal{D}, H)$.

Definition 5.19 (Well-behaved h-world)**.** Let $\mathcal{D}$ be any EPEC domain description. An h-world (W, H) is said to be *well-behaved w.r.t.* $\mathcal{D}$ if and only if the following two conditions are satisfied: i) it satisfies CWAS, ii) W is well-behaved w.r.t. $\mathcal{D}_H$.

Definition 5.20 (Model)**.** We extend the concept of a model to h-worlds in the following way. A *model of a domain description* $\mathcal{D}$ is a function such that, if (W, H) is well-behaved w.r.t. $\mathcal{D}$ then

$$M_{\mathcal{D}}(W, H) = \tilde{M}_{\mathcal{D}}^{\mathcal{D}_H}(W, H) \tag{5.4}$$

for $\mathcal{D}_H$ defined as in Definition 5.19, and equals 0 otherwise.

Proposition 5.3 (Model is a Probability Distribution)**.** Let $\mathcal{D}$ be an EPEC domain description. Then, $\mathcal{M}_{\mathcal{D}}$ is a probability distribution in the sense that it satisfies eq. (2.4) from section 2.2.1.

Proof. The proof is similar to that of corollary 5.1.2, since

$$\mathcal{M}_{\mathcal{D}}(W, H) = \tilde{M}_{\mathcal{D}}^{\mathcal{D}_H}(W, H) = M_{\mathcal{D}_H}(W) \cdot e_{\mathcal{D}}(H \mid W)$$

with both $M_{\mathcal{D}_H}$ and $e_{\mathcal{D}}(\cdot \mid W)$ being probability distributions for any $W \in \mathcal{W}$. □

Example 5.9. Using the results from example 5.8, $M_{\mathcal{D}_L}$ is defined as follows:

$$M_{\mathcal{D}_L}(W,H) = \begin{cases} \tilde{M}^{\mathcal{D}'_L}_{\mathcal{D}_L}(W,H) & \text{when } H = H_1 \\ \\ \tilde{M}^{\mathcal{D}''_L}_{\mathcal{D}_L}(W,H) & \text{when } H = H_2 \end{cases}$$

The following definition introduces b-propositions, which are analogous to h-propositions in PEC+ and are entailed by EPEC domain descriptions.

Definition 5.21 (b-proposition)**.** A b-proposition has the form

$$\textbf{at } I \textbf{ believes } \varphi \textbf{ with-probs } \{([H_1]_{<I}, B_1, P_1), \ldots, ([H_m]_{<I}, B_m, P_m)\}$$

for some instant I, i-formula φ, equivalence classes $[H_1]_{<I}, \ldots, [H_m]_{<I}$, reals $B_1, P_1, \ldots, B_m, P_m \in [0,1]$ such that $\sum_{i=1}^{m} B_i = 1$.

Definition 5.22 (Entailment for Domain Descriptions)**.** Given a domain description $\mathcal{D}$ and an i-formula φ, it is said that *the b-proposition "* **at** I **believes** φ **with-probs** $\{([H_1]_{<I}, B_1, P_1), \ldots, ([H_m]_{<I}, B_m, P_m)\}$*" is entailed by* $\mathcal{D}$ iff $M_{\mathcal{D}}(\varphi \mid [H_i]_{<I}) = P_i$ and $M_{\mathcal{D}}([H_i]_{<I}) = B_i$ for $i = 1, \ldots, m$.

Intuitively, if a domain description $\mathcal{D}$ entails a b-proposition

$$\textbf{at } I \textbf{ believes } \varphi \textbf{ with-probs } \{([H_1]_{<I}, B_1, P_1), \ldots, ([H_m]_{<I}, B_m, P_m)\}$$

this means that at instant I the agent (according to the information in the domain $\mathcal{D}$) believes that the i-formula φ holds with one of the probabilities $P_1, \ldots, P_m$ depending on the result from its sensors, which is "recorded" in $[H_i]_{<I}$ which has an associated probability B_i of being actually experienced.

5.3 Example Entailments

Example 5.10. The domain description $\mathcal{D}_L$ from example 5.2 entails, among others, the following b-propositions:

($|\!=$L1) **at** 0 **believes** $[\mathit{Light} = \mathit{On}]@0$ **with-probs** $\{(\langle\rangle, 1, 0.5)\}$

($|\!=$L2) **at** 0 **believes** $[\mathit{Light} = \mathit{On}]@2$ **with-probs** $\{(\langle\rangle, 1, 0.8625)\}$

($|\!=$L3) **at** 0 **believes** $[\mathit{Light} = \mathit{Off}]@2$ **with-probs** $\{(\langle\rangle, 1, 0.1375)\}$

($|\!=$L4) **at** 1 **believes** $[\mathit{Light} = \mathit{On}]@1$ **with-probs**
$\{(\langle(\mathit{sign}(s), \mathit{Light} = \mathit{On})@0\rangle, 0.55, 0.8\overline{63}),$
$\{(\langle(\mathit{sign}(s), \mathit{Light} = \mathit{Off})@0\rangle, 0.45, 0.0\overline{5})\}$

($|\!=$L5) **at** 2 **believes** $[\mathit{Light} = \mathit{On}]@2$ **with-probs**
$(\langle(\mathit{sign}(s), \mathit{Light} = \mathit{On})@0\rangle, 0.55, 0.8\overline{63}),$
$(\langle(\mathit{sign}(s), \mathit{Light} = \mathit{Off})@0\rangle, 0.45, 0.86\overline{1})\}$

where s is the only s-proposition in $\mathcal{D}_L$, i.e. *L5*.

Among these entailments, $|\!\models$*L2* is of particular interest. Indeed, if having the light turned on at instant 2 is considered to be a "goal" (from a planning perspective), this b-proposition shows that this goal has probability 0.8625, which is a significant improvement over the 50% chance of having the light turned on initially.

5.4 Summary

This chapter introduces EPEC, which constitutes the main contribution of this thesis. EPEC builds upon PEC+, and therefore has a similar syntax, and augments its possible-worlds semantics with *sensing histories* to model sensing actions. A *fixpoint semantics* is also introduced, which allows for modelling knowledge-conditioned actions in a probabilistic setting. Similar properties to those of PEC+ are proved, i.e. that the model function is also a probability function in this setting.

Chapter 6

Approximate Computation of h-props

As demonstrated in chapter 4, the provided ASP implementation of PEC+ performs temporal projection in a reasonable time for some domains. However, when an exact computation of the results is intractable, other strategies can be employed. This chapter deals with approximating the result of a given query using a probabilistic programming language.

6.1 Approximating h-propositions

Sections 4.3.4 and 4.4 show, through a proof of correctness, that if a domain description $\mathcal{D}$ entails an h-proposition of the form

$$\varphi \textbf{ holds-with-prob } P \tag{6.1}$$

then the provided ASP implementation outputs the value P on query φ. However, this implementation is not suitable for those domains where the number of worlds grow in such a way as to make automated reasoning intractable. In these cases, alternative techniques might be used which trade-off computation time for precision. A largely applied technique is that of *sampling*, i.e. extracting a set of samples from a given *population* which are then used to estimate some *parameters* (e.g., mean, standard deviation) of the population itself that are difficult to derive analytically. In PEC+ the main reasoning task is estimating the probability P such that proposition (6.1) is entailed by some domain of interest $\mathcal{D}$. In this case, the probability P is the parameter to be estimated through sampling from the population of well-behaved worlds w.r.t. $\mathcal{D}$. Let $\hat{W}_1, \ldots, \hat{W}_m$ be sampled well-behaved worlds w.r.t. $\mathcal{D}$ that are drawn from the probability distribution $M_{\mathcal{D}}$ (possibly with repetitions). The probability of sampling a world $\hat{W}$ which satisfies the i-formula φ (i.e. $\hat{W} \mid\!\models \varphi$) is P. Therefore, the process of sampling m worlds and testing them for satisfaction of φ can be modelled using m independent Bernoulli random variables with success probability P, i.e. random variables $X_1, \ldots, X_m$ such that

$$X_i = \begin{cases} 1 & \text{with probability } P \\ 0 & \text{with probability } 1 - P \end{cases}$$

for every $i = 1, \ldots, m$. The probability of getting n successes among these m trials is given by the random variable $Y = X_1 + \cdots + X_m$ which is binomially distributed with parameters P (the success probability) and m (the number of trials). It can be shown that Y has expected value mP. Therefore, if P is estimated by the quantity

$$\hat{P} = Y/m \tag{6.2}$$

then the expected value for $\hat{P}$ is $mP/m = P$, i.e. $\hat{P}$ is an *unbiased* estimator for P. A standard result from probability theory states that when m (the number of trials) is big enough, the distribution of expected values for $\hat{P}$ is well approximated by a normal distribution with mean P and standard deviation $\sqrt{P(1-P)/m}$, meaning that the probability of getting an estimate that deviates significantly from P decreases as m gets larger. , under this assumption of normality, there is a 95% probability that the estimate $\hat{P}$ falls in the interval $[P - \epsilon_m, P + \epsilon_m]$ for $\epsilon_m = 1.96\frac{P(1-P)}{m}$, and notice that $\epsilon_m \to 0$ when $m \to \infty$. Vice-versa, given sampled worlds $\hat{W}_1, \ldots, \hat{W}_m$ and the associated estimates $\hat{P}$ and $\hat{\sigma} = \sqrt{\hat{P}(1-\hat{P})/m}$ one can calculate a *confidence interval* for P as $[\hat{P} - z\hat{\sigma}, \hat{P} + z\hat{\sigma}]$, where z depends on the required *confidence level* (e.g. 95%). This means that one can be *confident* that the actual value P can be found in the interval $[\hat{P} - z\hat{\sigma}, \hat{P} + z\hat{\sigma}]$ with that confidence level.

Confidence intervals are introduced in PEC+ through the following definitions, which constitute a progressive weakening of definition 3.34:

Definition 6.1 (Interval Entailment for Domain Descriptions)**.** Given a domain description $\mathcal{D}$, and an i-formula φ, it is said that *the h-proposition "*φ **holds-with-prob** $[P, P']$*" is entailed by* $\mathcal{D}$ if $\exists P'' \in [P, P']$ such that $\mathcal{D} \mid\!\models \varphi$ **holds-with-prob** P'', and write

$$\mathcal{D} \mid\!\models \varphi \textbf{ holds-with-prob } [P, P']$$

By definition, if some domain description $\mathcal{D}$ is such that $\mathcal{D} \mid\!\models \varphi$ **holds-with-prob** $[P, P']$, then one can be 100% confident that the "true" probability P'' lies within the interval $[P, P']$, as this relies on the strict logical entailment of the "true" h-proposition $\mathcal{D} \mid\!\models \varphi$ **holds-with-prob** P. However, as it was shown in previous paragraphs, when probabilistic approximation techniques are employed instead of strict logical entailment, 100% confidence intervals would always degenerate into $[0, 1]$ intervals. This motivates an even weaker definition:

Definition 6.2 (Confidence Interval Entailment for Domain Descriptions)**.** Given a domain description $\mathcal{D}$, an i-formula φ, a confidence level c, an approximation technique a producing confidence intervals at the confidence level c, *the h-proposition "*φ **holds-with-prob** $[P, P']$*" is entailed by* $\mathcal{D}$ *at the confidence level* c *by the approximation technique* a if a run of the approximation technique produces $[P, P']$ as a confidence interval for the probability P'' such that $\mathcal{D} \mid\!\models \varphi$ **holds-with-prob** P''. This is also written

$$\mathcal{D} \mid\!\models_{(c,a)} \varphi \textbf{ holds-with-prob } [P, P']$$

The shorthand

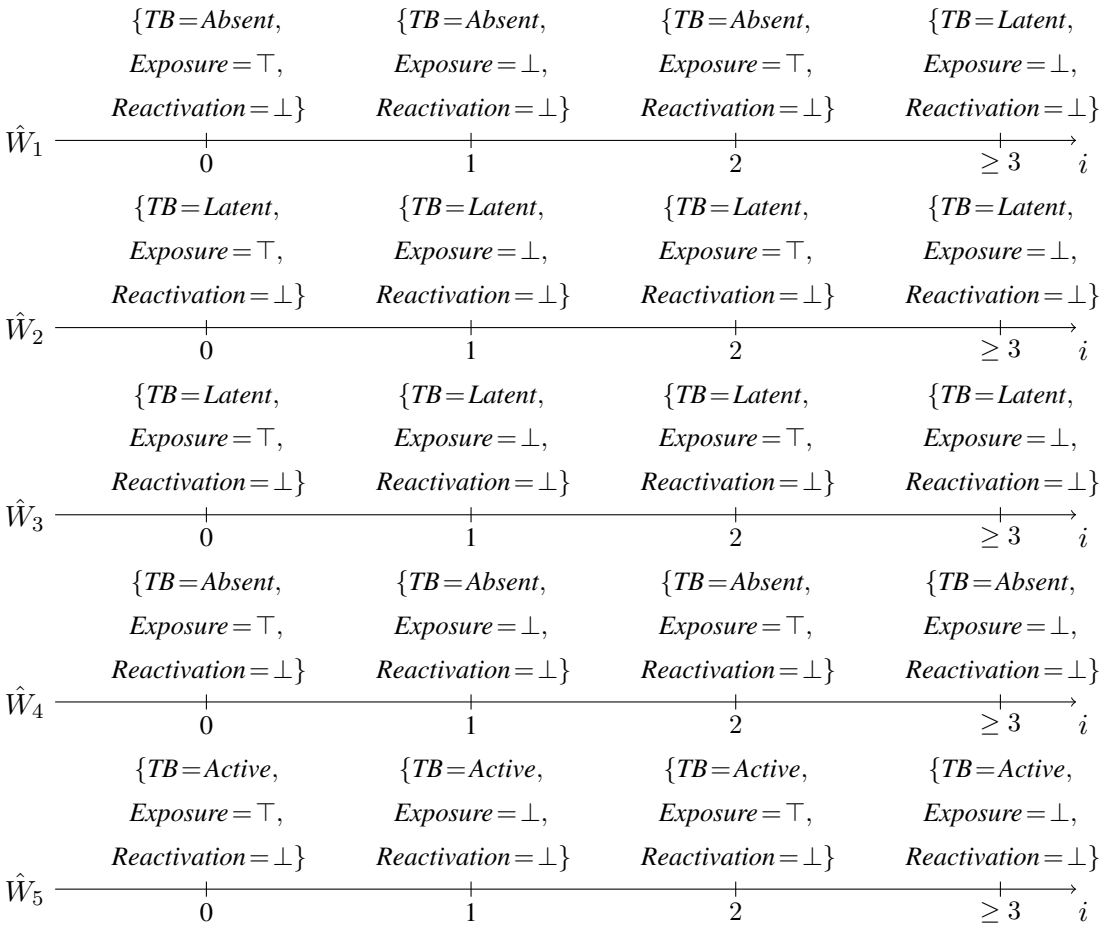
$$\mathcal{D} \mid\!\models_c \varphi \textbf{ holds-with-prob } [P, P']$$

can be used when a is implicit.

Example 6.1. Table 4.1 shows that the tuberculosis domain description $\mathcal{D}_T$ as in example 3.8 entails

$$[\mathit{TB}{=}\mathit{Active}]@50 \textbf{ holds-with-prob } 0.0893061 \tag{6.3}$$

and consider a sampling procedure that produces the following worlds:



In this example, only one world out of five generated satisfies the i-formula $[\mathit{Tuberculosis}{=}\mathit{Active}]@50$, therefore this procedure approximates the correct probability $M_{\mathcal{D}_T}([\mathit{Tuberculosis}{=}\mathit{Active}]@50) = 0.0893061$ by $1/5 = 0.2$. As an example, a 95% confidence range for the "true" probability can be calculated in this case as $[\max(0.2 - 1.96 \cdot \sqrt{0.2 \cdot 0.8/5}, 0), \min(0.2 + 1.96 \cdot \sqrt{0.2 \cdot 0.8/5}, 1)] \approx [0, 0.55]$, but notice that this is unreliable as the number of sampled world is small ($m = 5$) and therefore the normal approximation is not a good one.

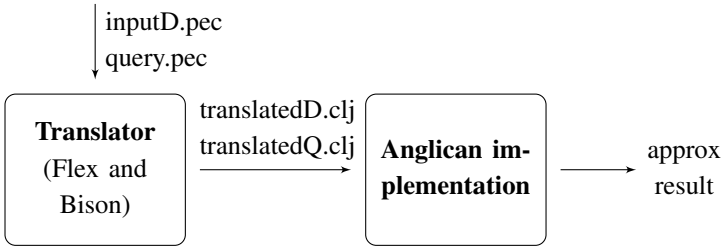


Figure 6.1: Schematic representation of the Anglican implementation of PEC+.

6.2 Architecture of the implementation

To produce more reliable estimates, more samples must be generated and to this aim some automated procedure is needed. In this thesis, this task is performed by Anglican [63], a recent probabilistic programming language based on Clojure (see e.g. [23]). Anglican performs this task from a specification of a probabilistic model (given by $M_{\mathcal{D}}$ in definition 3.32 in the context of this thesis) and then applying one of its several built-in techniques. The architecture of the implementation is similar to the ASP one (see fig. 6.1). Flex and Bison transform a PEC+ domain description (compatible with the ASP implementation) and a (conditional) query into an Anglican-readable format. This is then given on input to an Anglican implementation whose task is that of creating a probabilistic model reflecting the input domain description and query, and then extracting a (user-specified) number of samples from that. An estimate $\hat{P}$ for the probability of the given query is then produced on output, from which confidence intervals can be calculated using the usual normality assumption and the associated formula. This implementation, currently under active development, is available at `https://github.com/dasaro/pec-anglican`. An implementation of EPEC based on PEC+'s Anglican implementation is also being developed and will be available at `https://github.com/dasaro/epec-anglican`.

6.3 Experiments

All the experiments below were generated using Monte Carlo for sampling, but Anglican readily allows for the use of other methods (e.g. Importance sampling, Markov Chain Monte Carlo). These experiments are used to provide some empirical evidence for the correctness of this implementation.

Example 6.2. As a first example, consider the simple Decay domain description from example 4.5. Figure 6.2 represent estimates for the probability of $[F = \top]@I$ for $I = 0, \ldots, 15$ calculated using Anglican and the corresponding 95% confidence ranges. The plot on the left refers to sampling 100 worlds whereas the plot on the right refers to sampling 1000 worlds. Notice how the number of samples impacts the confidence ranges. "True" probabilities were calculated using the ASP implementation. Also note that in both cases, $14/15$ times the "true" value falls within the 95% confidence intervals. Clearly, even more precision can be achieved if even more worlds are sampled.

Example 6.3. Figure 6.3 show estimates and true probabilities (calculated using ASP) for the

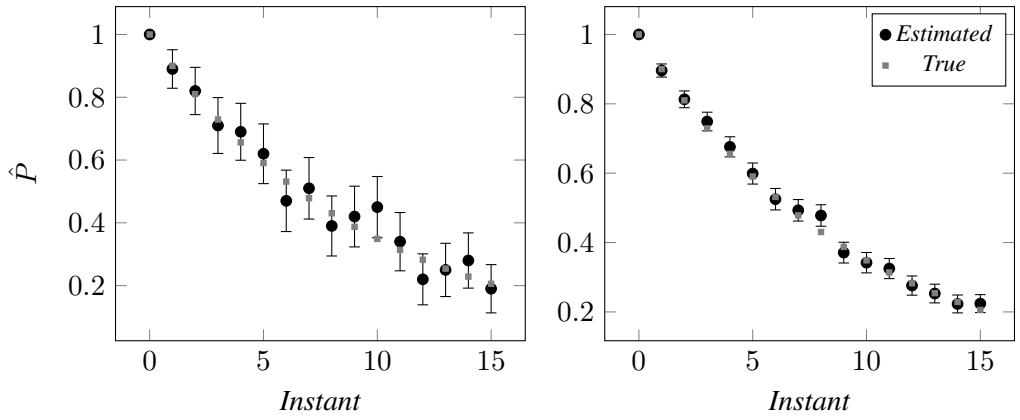


Figure 6.2: A graphical representation of the confidence intervals $[P, P']$ *such that* $\mathcal{D}_D \ \|\!\models_{95\%}$ $[F = \top]@I$ **holds-with-prob** $[P, P']$ *for* $I = 0, \ldots, 15$*. While the plot on the left refers to sampling* 100 *worlds, the plot on the right refers to sampling* 1000 *worlds. The true value, calculated using ASP, is also plotted for reference.*

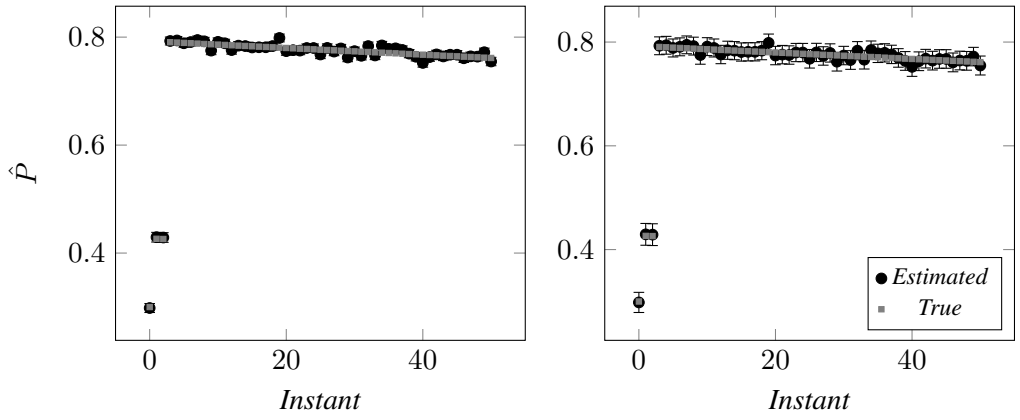


Figure 6.3: The plot on the left represents confidence intervals $[P, P']$ *such that* $\mathcal{D}_L \ \|\!\models_{95\%}$ $[\mathit{Tuberculosis} = \mathit{Latent}]@I$ **holds-with-prob** $[P, P']$ *for* $I = 0, \ldots, 50$*. The plot on the right represents confidence intervals* $[P, P']$ *such that* $\mathcal{D}_L \ \|\!\models_{99.73\%}$ $[\mathit{Tuberculosis} = \mathit{Latent}]@I$ **holds-with-prob** $[P, P']$ *for* $I = 0, \ldots, 50$*. The true value, calculated using ASP, is also plotted for reference.*

probability of $[\mathit{Tuberculosis} = \mathit{Latent}]@I$ for $I = 0, \ldots, 50$ in the Tuberculosis domain description from example 3.8. The same set of samples of cardinality 5000 is shown in both figures, but plotted confidence ranges are 95% for the figure on the left and 99.73% for the one on the right. Notice that while interval estimates on the left are very small, ones on the right are bigger as to guarantee a 99.73% confidence level.

For what regards computational time, every query requires a pre-processing time mainly depending on the size of the domain description, and a sampling time which linearly depend on the number of generated samples. It is important to note that computational time does *not* depend on the number of traces like in the ASP implementation. Some results about average sampling time on different domain description is shown in fig. 6.4. Loading time was found to be constant in both the tuberculosis and decay scenario, with an average loading time of $0.15\,\mathrm{sec}$ for the Decay domain description and $0.66\,\mathrm{sec}$ for the tuberculosis scenario.

This suggests that the Anglican implementation can be employed to approximate the correct answer to a query even when the ASP implementation does not provide an answer for compu-

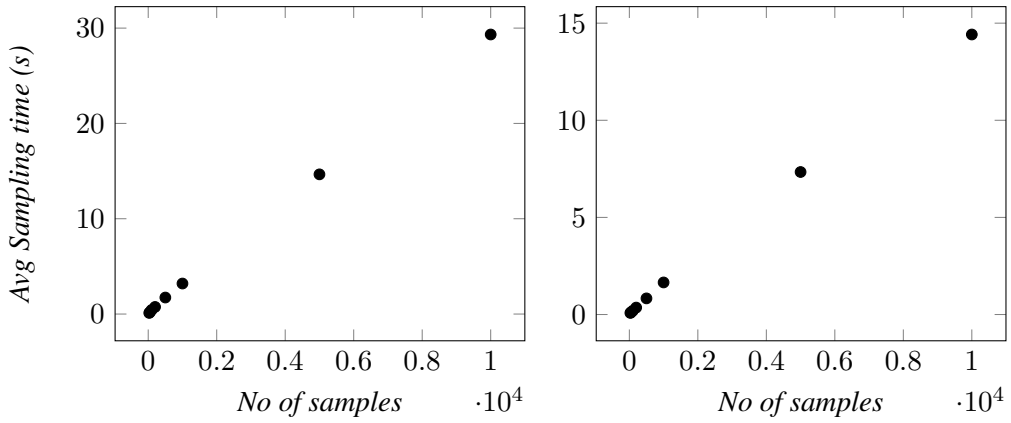


Figure 6.4: Average time spent on generating a given number of samples using different domain descriptions. The plot on the left refers to the Decay domain description, while the plot on the right refers to the Tuberculosis domain description.

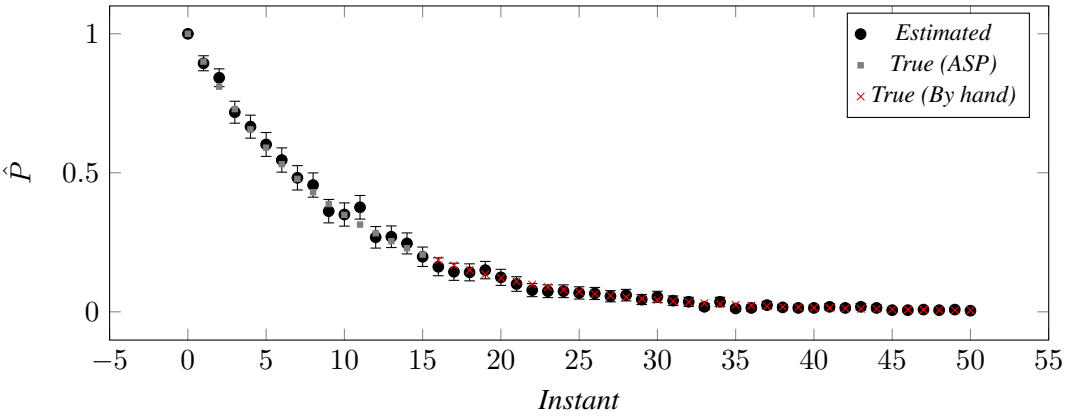


Figure 6.5: A graphical representation of the confidence intervals $[P, P']$ *such that* $\mathcal{D}_D \models_{95\%} [F = \top]@I$ **holds-with-prob** $[P, P']$ *for* $I = 0, \ldots, 50$. 500 *samples were generated for each query.*

tational complexity reasons. This is demonstrated in the following example.

Example 6.4. In the Decay experiment (see fig. 4.5) it was shown that ASP can take up to about 1 hour to provide an answer to some queries. However, Anglican does not suffer from the same problem and therefore can be used in cases where exact reasoning is intractable. In fig. 6.5, 95% confidence intervals for the probability of $[F = \top]@I$ for $I = 0, \ldots, 50$ are shown based on an Anglican-generated set of 500 samples. "True" values are again calculated using ASP but are unavailable for $I > 15$ due to complexity reasons, so the ground truth was calculated by hand when $I > 15$. It is worth noting that, in Anglican, every query required on average $1.732\,\text{sec}$ to be performed.

Chapter 7

Related Work

This chapter describes some recent languages for RAA that also support some form of probabilistic reasoning. It provides a high-level overview of these languages and compares them to PEC+ and EPEC by commenting on their similarities and differences. While Situation Calculus-based formalisms have a stronger tradition and generally focus on planning (as already mentioned in sections 2.1.2 and 2.1.3 and fig. 2.1), Event Calculus-based formalisms are more recent and oriented to dealing with narratives. This chapter is partitioned into two section so as to reflect these two approaches.

7.1 Situation Calculus formalisms

7.1.1 BHL

The *BHL* framework [3] (here named after its authors: Bacchus, Halpern and Levesque) is perhaps the first attempt to integrate a language for RAA with some model of uncertain and epistemic reasoning. BHL is based on Reiter's formulation of the Situation Calculus discussed in section 2.1.2 and therefore inherits its ontology and is formulated in terms of second-order logic axioms.

Just like EPEC, this framework is general enough to support different models of belief such as probabilistic, possibilistic or Dempster-Schafer belief functions, but just like in this thesis its authors chose to focus on the probabilistic case. Their way of encoding probabilities is by using a functional fluent $p(s', s)$ that, for any two situations s and s', represents the degree of belief that an agent in situation s assigns to the situation s' being the actual situation. Similarly to the extension of a model to a function over formulas in EPEC, an appropriate function $BEL(\theta, s)$ over formulas is then defined on top of the functional fluent p as the normalised sum of weights of those situations in which θ holds from the point of view of an agent in situation s. This is then shown to satisfy the axioms of probability.

BHL can model both noisy *effectors* (i.e., actions whose effects are uncertain) and noisy *sensors* (i.e., sensing actions whose outcomes are imperfect). These loosely correspond to c-propositions and s-propositions in EPEC respectively. As an illustration, consider a simple scenario in which objects can be dropped. As a result of dropping there is a chance they will break if they are fragile. This is implemented in BHL though the following set of axioms:

(BHL-D1) $\mathit{Poss}(\mathit{DropBreak}(x), s) \leftrightarrow \mathit{Holding}(x, s) \wedge \mathit{Fragile}(x, s)$

(BHL-D2) $\mathit{Poss}(\mathit{DropNotBreak}(x), s) \leftrightarrow \mathit{Holding}(x, s)$

(BHL-D3) $l(\mathit{DropBreak}(x), s) =$ **if** $\mathit{Fragile}(x)$ **then** 0.8 **else** 0

(BHL-D4) $l(\mathit{DropNotBreak}(x), s) =$ **if** $\mathit{Fragile}(x)$ **then** 0.2 **else** 1

Axioms *BHL-D1* and *BHL-D2* define the precondition of the two actions *DropBreak* and *DropNotBreak*, whose effect is governed by the successor state axiom

$$\mathit{Poss}(a, s) \rightarrow \mathit{Broken}(x, \mathit{Do}(a, s)) \leftrightarrow a = \mathit{DropBreak}(x) \vee \mathit{Broken}(x, s)$$

and axioms *BHL-D3* and *BHL-D4* state what the likelihoods of success of the two actions are. The agent cannot execute these two actions directly, he can only perform an action $\mathit{Drop}(x)$ which non deterministically activates one of $\mathit{DropBreak}(x)$ or $\mathit{DropNotBreak}(x)$. This is loosely equivalent to the following c-proposition in PEC+:

(PEC-D1) $\mathit{Drop} \wedge \mathit{Holding} = \top \wedge \mathit{Fragile} = \top$ **causes-one-of** $\{(\{\mathit{Broken} = \top\}, 0.8)\}$

Although PEC+ allows for a more compact representation (this is often the case when comparing action languages with classical logic formalisms), BHL is a second order formalism and can therefore use a more expressive language e.g. to express parametrised actions such as $\mathit{Drop}(x)$. This can also be simulated in PEC+ as long as the parameter takes value in a finite domain.

However, BHL cannot represent narratives and therefore there is no p-proposition or o-proposition equivalent, as the main focus of this work was exploring how noisy effectors and sensors interact and impact the agent's belief as a result. This work has since then served as an inspiration for other similar frameworks, see e.g. [43], and has been further developed in many ways. For instance, [6] adds support for continuous probability distributions, and [7] applies BHL and these extensions to to the problem of localisation, i.e. to the case where an agent moves in a multi-dimensional world and can sense its position. The problem of extending BHL with a modality known as *only knowing*, which allows for the modelling of a precise specification of what is and what is *not* known within a logical theory of actions, has also been tackled in [5].

7.1.2 Language PAL

Language PAL (for *Probabilistic Action Language*) is a probabilistic extension of Language $\mathcal{A}$, which is described in section 2.1.4. It is an action language, and, just like Language $\mathcal{A}$, is purposely simple. A distinguishing feature of PAL is that it is *elaboration tolerant*, i.e. new knowledge can be added to domain descriptions without having to revisit them entirely. This feature is usually overlooked in probabilistic RAA languages (including PEC+ and EPEC) and is a consequence of how probabilities are represented in PAL. As an example, c-proposition *T3* from example 3.8 can be translated to PAL in the following way:

(PAL-T3.1) **probability of** A **is** $4/100$

(PAL-T3.2) **probability of** L **is** $76/100$

(PAL-T3.3) *Exposure* **causes** *Active*, ¬*Latent*, ¬*Absent* **if** *Absent*, A

(PAL-T3.4) *Exposure* **causes** ¬*Active*, *Latent*, ¬*Absent* **if** *Absent*, ¬A, L

This example can be used to illustrate some characteristics of PAL. The two propositions *PAL-T3.1* and *PAL-T3.1* are called *Probability Description Propositions* and define two random variables A and L and their associated probability distributions. Random variables in PAL are similar to fluents (which however cannot be affected by actions) and are of two kinds: inertial and non-inertial. In both cases their value is picked at random according to the specified distribution, with the difference that inertial random variables persist whereas non-inertial random variables do not, and therefore their values are re-sampled whenever a new situation is reached. Random variables defined through Probability Description Propositions are always assumed to be independent from each other. In this example, it makes sense for both A and L to be non-inertial random variables.

Propositions *PAL-T3.3* and *PAL-T3.4* are called *Dynamic Causal Laws* and specify what the effect of an action is under some (possibly probabilistic) preconditions. Since PAL is non-functional, in this example the three possible values for *Tuberculosis* are represented as separate boolean fluents. In this example, the two effects of *Exposure* (i.e., *Active*, ¬*Latent*, ¬*Absent* and ¬*Active*, *Latent*, ¬*Absent*) are probabilistic as the preconditions of propositions *PAL-T3.3* and *PAL-T3.4* depend not only on the fluent *Absent* but also on the two random variables A and L and therefore on their sampled values.

Now suppose that one wanted to enrich the domain description by adding a new outcome for the action *Exposure* to account for the possibility of the patient developing a resistance to tuberculosis after being exposed to it. In PAL this would require *adding* the following axioms:

(PAL-T3.5) **probability of** I **is** $1/100$

(PAL-T3.6) *Exposure* **causes** *Immune* **if** *Absent*, ¬L, ¬A, I

whereas in PEC+ one would have to *change* proposition *T3* to

(T3*) $\mathit{Exposure} \wedge \mathit{Tuberculosis} = \mathit{Absent}$
causes-one-of
$\{(\mathit{Tuberculosis} = \mathit{Active}, 4/100),$
$(\mathit{Tuberculosis} = \mathit{Latent}, 76/100),$
$(\mathit{Immune}, 1/100),$
$(\emptyset, 19/100)\}$

which shows how PAL is fairly elaboration tolerant when compared to PEC+.

PAL also deals with *ramifications* (i.e., knock-on effects of actions) through *static causal laws* of the form

$$\psi \textbf{ causes } \phi$$

for a fluent formula ψ and a formula ϕ of fluents and inertial variables.

PAL can also handle *executability conditions* through propositions of the form

$$\textbf{impossible } A \textbf{ if } \theta$$

for an action A and a formula θ of fluent and unknown variables. Both static causal laws and executability conditions cannot be expressed in PEC+ and EPEC.

To illustrate PAL's semantics, it is convenient to use a simple ball-drawing example. Consider the following domain description:

(PBD1) *AttemptDraw* **causes** *Red* **if** U,

(PBD2) *AttemptDraw* **causes** $\neg$*Red* **if** $\neg U$,

(PBD3) **probability of** U **is** $1/4$.

This domain description describes an experiment in which balls are drawn with replacement from an urn, and its interpretation changes according to whether U is inertial or non-inertial. If U is inertial, the balls in the urn are all of the same colour, either blue (with probability $3/4$) or red (with probability $1/4$). If U is non-inertial, the balls in the urn are red and blue in proportion $1:3$.

Similar to many other situation calculus RAA frameworks, the semantics of PAL is given in terms of *states* and *transitions* and builds a probability function P that is similar to the concept of a model in PEC+. In a nutshell, *states* are interpretations of every fluent and variable in the language. In the ball-drawing example these are:

$$\{\textit{Red}, U\}, \{\neg\textit{Red}, U\}, \{\textit{Red}, \neg U\}, \{\neg\textit{Red}, \neg U\}$$

The probability P of a given state is then calculated on the basis of (domain-dependant) probabilities of unknown variables and assuming that fluent states (that is, interpretations of all fluents in the language) are equiprobable to each other. In the example above $P(U) = 1/4$, $P(\neg U) = 3/4$, $P(\textit{Red}) = P(\neg\textit{Red}) = 1/2$ and

$$P(\{\textit{Red}, U\}) = P(\{\neg\textit{Red}, U\}) = 1/8,\ P(\{\textit{Red}, \neg U\}) = P(\{\neg\textit{Red}, \neg U\}) = 3/8$$

The *transition* from a state to another represents the probability of transiting from one state to another due to a sequence of actions. It depends only on the non-inertial variables and the effect of actions on fluents (since inertial variables cannot change their values from a state to another). In the ball-drawing example, if U is inertial a transition from $\{\neg\textit{Red}, \neg U\}$ to $\{\textit{Red}, U\}$ due to *AttemptDraw* is given probability 0 because it is not possible for a inertial variable to change value. If U is non-inertial this same transition is given probability $1/4$.

PAL uses *queries* to express what is the probability of a formula after the execution of a sequence of actions. They are of the form

$$\textbf{probability of } [\psi \textbf{ after } A_1, \ldots, A_n] \textbf{ is } p$$

where ψ is a formula of fluents and unknown variables. A query of this form is *entailed* by a domain description iff p equals the sum of probabilities of a transition from any initial state s to another state in which ψ holds after actions $A_1, \ldots, A_n$ are executed, weighted on the

probability $P(s)$ of starting from state s. In the ball drawing example, the query

$$\textbf{probability of } [\textit{Red} \textbf{ after } \textit{AttemptDraw}, \ldots, \textit{AttemptDraw}] \textbf{ is } 1/4$$

is entailed by the corresponding domain description (regardless of U being inertial or non-inertial).

Finally PAL can also deal with *hypothetical observations*, and perhaps even more interestingly it is also able to deal with *narratives*.

Hypothetical observations in PAL have the form

$$\psi \textbf{ obs-after } A_1, \ldots, A_n.$$

and are hypothetical in the sense that they did not really happen. Real observations are dealt with in a narrative extension of the language that, building upon [48], extends PAL with time-points and allows for the expression of propositions of the form

$$\phi \textbf{ at } t$$

and

$$\alpha \textbf{ occurs-at } t$$

for a fluent formula ϕ, a (possibly empty) sequence of actions α and time-point t.

Although they are based on different ontologies, PAL and PEC+ are on a par for what regards representation of narratives, the only difference being that PEC+ can handle probabilistic event occurrences (of the form "A **happens-at** I **with-prob** P^+") while PAL cannot. However, unlike in EPEC, there is no way to model sensing actions in PAL.

7.1.3 Language $\mathcal{E}+$

Language $\mathcal{E}+$ [33], not to be confused with language $\mathcal{E}$ discussed in section 2.1.4, is a probabilistic action language with an epistemic component. It is mainly based on the action languages $\mathcal{C}+$ [30] and $P\mathcal{C}+$ [18]. Its distinguishing aspect is that it is able to express both probabilistic and non-deterministic actions, alongside (perfect) sensing actions. As many situation calculus formalisms the main focus of $\mathcal{E}+$ is put on planning. Since it does not support imperfect sensing (unlike PEC+ and EPEC), $\mathcal{E}+$ is suitable for domains in which it is reasonable to assume that sensing actions produce perfect knowledge about the world. This shows a crucial difference in focus between $\mathcal{E}+$ and the frameworks proposed in this thesis, as some domains that EPEC is able to model (see e.g. scenarios 1.2 to 1.4) cannot be realistically modelled in $\mathcal{E}+$. In the remainder of this section, some characteristics of its syntax and semantics are outlined.

Its basic language is made of *fluents* and *actions*, with the latter being partitioned into *physical actions* and *sensing actions*. Physical actions are further divided into *deterministic*, *non deterministic* and *probabilistic* actions.

It allows for the definition of several types of propositions: *precondition axioms* are used to express the conditions under which a (physical or sensing) action is executable. *Conditional effect axioms* express the effects of an action when the world satisfies certain conditions; in

particular, non deterministic effect axioms specify the possible outcomes of that physical action, while probabilistic effect axioms attach a probability to each effect, which are considered mutually exclusive. *Sensing effect axioms* are used to specify which literals a sensing action produces knowledge about. *Default frame axioms* serve to specify which features of the world persist when a particular action is executed. Finally, *domain constraint axioms* are used to describe background knowledge which is invariant to the execution of an action.

The semantics of $\mathcal{E}+$ is based on the concept of a *state* and *epistemic state*: as in other languages described in this thesis, a state is an interpretation of fluents in the language, while an epistemic state is a set of states representing what the agent thinks it might be true in the actual world: for instance, the epistemic state $\{\{F, \neg G\}, \{F, G\}\}$ can be used to represent ignorance about the truth value of G. It plays a role analogue to that of the Epistemic Fluent $K(w)$ in EFEC, with a significant difference: an Epistemic State can only represent the instantaneous accessibility to a set of states (i.e., no temporal information is taken into account), while the Epistemic Fluent represents an accessibility relation to entire worlds, which bear information about events and when they take place.

The epistemic part of $\mathcal{E}+$ is implemented through a directed graph $G = \langle V, E\rangle$ where the set V of vertices consists of all the epistemic states and there is an edge from S to S' labelled α if and only if the action α is executable on the epistemic state S (meaning that S satisfies all the Precondition Axioms) and produces S' as a result. In the case where α is a physical action, the result of executing it on S will produce a new epistemic state S' (called *successor epistemic state of* S) where all the following conditions are satisfied: i) S' includes all the physical effects of α mentioned in the corresponding Conditional Effect Axioms whenever their preconditions are satisfied by the current epistemic state S, ii) S' satisfies all the indirect effects encoded in the Domain Constraint Axioms, iii) S' satisfies the inertial constraints encoded in the Default Frame Axioms. The case of α being a sensing action is similar: instead of taking into account physical effects, the successor epistemic state has to be consistent with the outcome of the corresponding sensing actions. Under the assumption that no probabilistic or non deterministic actions are in the domain description, such a successor epistemic state, if one exists, is unique

Notice that since the graph G includes only epistemic states, all the preconditions are to be interpreted as *epistemic preconditions*, that is, it does not matter whether a fluent precondition is true in the actual environment: it is impossible to execute an action until such this precondition is *known* to hold.

Every non deterministic and probabilistic effect axiom is to be considered (logically or probabilistically) independent of every other axiom, so if we are given an action α, an epistemic state S and the maximal set of Probabilistic Conditional Effect Axioms

$$\begin{gathered}\textbf{caused } \psi_{1,1} : p_{1,1}, \ldots \psi_{1,n_1} : p_{1,n_1} \textbf{ after } \alpha \textbf{ when } \phi_1, \\ \textbf{caused } \psi_{2,1} : p_{2,1}, \ldots \psi_{2,n_2} : p_{2,n_2} \textbf{ after } \alpha \textbf{ when } \phi_2, \\ \ldots, \\ \textbf{caused } \psi_{k,1} : p_{k,1}, \ldots \psi_{k,n_k} : p_{k,n_k} \textbf{ after } \alpha \textbf{ when } \phi_k\end{gathered}$$

in the domain description such that all the ϕ_i are satisfied in S, the semantics calculates the probability that α will trigger a transition from S to a new epistemic state such that exactly one

effect ψ_{i,j_i} has been picked for each $i = 1, 2, \ldots, k$ as $\sum \prod_{i=1}^{k} p_{i,j_i}$, where the sum is over all possible choices of effects that lead to S' (clearly, if the axioms are non deterministic instead, no probability is calculated). Conversely, all the successor epistemic states of a state S take the form of a state S' which has been formed from S by taking into account inertia, indirect effects, and some combination of direct effects.

Typically, an initial state description δ_I, in the form of a fluent conjunction, is also taken into account: semantically speaking, this has the effect of restricting the graph $G = \langle V, E \rangle$ to a graph $G_{\delta_I} = \langle V_{\delta_I}, E_{\delta_I} \rangle$ where only successor epistemic states of the initial epistemic state are considered.

In [33], the authors focus on developing algorithms for planning using $\mathcal{E}+$. They define the *quality* of a conditional plan (in the form of a sequence of actions) as the lower probability that such a plan will reach one of the epistemic states satisfying a desirable *goal*, typically a fluent conjunction. An algorithm to generate an optimal plan (i.e., such that its probability is minimal) is also introduced and discussed.

7.2 Event Calculus Formalisms

7.2.1 MLN-EC and Prob-EC

As mentioned earlier, the Event Calculus had not been merged with some form of probabilistic reasoning until recently. The first attempt to do so is *MLN-EC* [62], followed by the closely related *Prob-EC* [61].

These two languages give a probabilistic semantics to EC, respectively using *Markov Logic Networks* (*MLN*, for short) and a recent probabilistic dialect of Prolog called *ProbLog* [14]. In the following, their characteristics are briefly discussed. Both of them are based on a discrete-time reworking of the EC that is close to that presented in section 2.1.3. As a case-study, they were both applied to the realisation of a system for recognizing human activity given a symbolic representation of video content from security cameras, but can be applied in general to the task of recognising a set of *Long-Term Activities* (*LTAs* for short) given as input a set of timestamped *Short-Term Activities* (STAs for short). They have both been shown to be particularly effective when dealing with event recognition in highly noisy environments, especially when compared to plain EC.

MLN-EC

The syntax of MLN-EC supports the assignment of weights to formulas in the knowledge base. This induces a ground MLN, which then defines a probability distribution over worlds accordingly. In MLN-EC, input weights are assigned to the causal axioms of EC. As an example, consider a simple theory with only one fluent F. Four implications can be weighted separately:

$$\textit{holdsAt}(F, T+1) \leftarrow [\textit{Initiation Conditions}]$$
$$\neg\textit{holdsAt}(F, T+1) \leftarrow \neg\textit{holdsAt}(F, T) \wedge \neg[\textit{Initiation Conditions}]$$
$$\neg\textit{holdsAt}(F, T+1) \leftarrow [\textit{Termination Conditions}]$$
$$\textit{holdsAt}(F, T+1) \leftarrow \textit{holdsAt}(F, T) \wedge \neg[\textit{Termination Conditions}]$$

where the initiation (resp. termination) conditions are user-defined.

Depending on which axiom is weighted, several behaviours can be obtained. If all of the axioms above are hard constrained (i.e., they are all assigned ∞ as a weight), the result is indistinguishable from plain EC. If the axioms that specify what holds after an initiation (resp. termination) are soft-constrained (i.e. they are assigned weight $< \infty$) then the initiation (resp. termination) of a fluent increases (resp. decreases) the probability of that fluent holding. This is similar to defining probabilities of c-proposition outcomes in PEC+. Soft-constraining persistence formulas generalise the law of inertia, so that the probability of that fluent holding might increase or decrease over time even when it is not initiated or terminated.

Prob-EC

Prob-EC's core, called Crisp-EC, is close in spirit to the original formulation of the Event Calculus as a logic program of [38]. Since ProbLog syntax is largely compatible with that of Prolog, only a few transformations had to be applied to Crisp-EC axioms in order to port it.

As a complete description of the system is out of the scope of this thesis, only the domain dependent part of their Crisp-EC theory devoted to recognising an LTA of type *Moving* (which is a fluent supposed to be true when two people are moving together) from a video stream is presented here:

$$\begin{aligned}\textit{InitiatedAt}(\textit{Moving}(P1, P2) = \textit{true}, T) \leftarrow \\ \textit{HappensAt}(\textit{Walking}(P1), T), \\ \textit{HappensAt}(\textit{Walking}(P2), T), \\ \textit{HoldsAt}(\textit{Close}(P1, P2, 34) = \textit{true}, T), \\ \textit{HoldsAt}(\textit{Orientation}(P1) = \theta_1, T), \textit{HoldsAt}(\textit{Orientation}(P2) = \theta_2, T), \\ |\theta_1 - \theta_2| < 45^\circ\end{aligned}$$

meaning that a *Moving* LTA is initiated whenever both people $P1$ and $P2$ are *Walking* (where *Walking* is an input STA), they are *Close* enough ($Close(P1, P2, 34)$ means that $P1$ and $P2$ are at most 34 pixels far from each other in the video frame) and they are heading roughly to the same direction (this is what the *Orientation* input predicate is for). The termination condition is given as

$$\begin{aligned} \textit{TerminatedAt}(\textit{Moving}(P1, P2) = \textit{true}, T) \leftarrow \\ \textit{HappensAt}(\textit{Walking}(P1), T), \\ \textit{HoldsAt}(\textit{Close}(P1, P2, 34) = \textit{false}, T) \end{aligned}$$

i.e. *Moving* is terminated when at least one of $P1$ and $P2$ is walking and they are not close to each other anymore.

An input flow of input STAs annotated with probabilities is then considered, such as the following:

$$\begin{aligned} &0.70 :: \textit{HappensAt}(\textit{Walking}(\textit{mike}), 1) \\ &0.46 :: \textit{HappensAt}(\textit{Walking}(\textit{sarah}), 1) \\ &0.73 :: \textit{HappensAt}(\textit{Walking}(\textit{mike}), 2) \\ &0.55 :: \textit{HappensAt}(\textit{Walking}(\textit{sarah}), 2) \\ &0.70 :: \textit{HappensAt}(\textit{Walking}(\textit{mike}), 3) \\ &0.38 :: \textit{HappensAt}(\textit{Walking}(\textit{sarah}), 3) \\ &0.61 :: \textit{HappensAt}(\textit{Walking}(\textit{mike}), 4) \\ &0.39 :: \textit{HappensAt}(\textit{Walking}(\textit{sarah}), 4) \\ &\ldots \end{aligned}$$

Notice how this is similar to assigning probabilities to p-propositions in PEC+.

The degree to which *Moving* holds at a given time point for Mike and Sarah is then calculated by combining all its previous initiations and terminations. For instance, in this case the probability of a Moving LTA at time 3 can be calculated as the probability that *Moving* has been initiated (and not terminated) at one of the previous time points: $\textit{HoldsAt}(\textit{Moving}(\textit{mike}, \textit{sarah}), 3)$ is then given probability equal to $0.70 \cdot 0.46 + 0.73 \cdot 0.55 - 0.70 \cdot 0.46 \cdot 0.73 \cdot 0.55 = 0.594$, under the assumption that they are *Close* and heading the same direction. A property of Prob-EC is that if no initiations or terminations for given fluent are in the input stream in a time period, then the probability that the fluent holds *persists through inertia*, i.e. it does not change. This is similar to the behaviour of PEC+ shown in fig. 3.2.

While Prob-EC relies on ProbLog for probabilistic inference, MLN-EC uses state-of-the-art approximation techniques (including e.g. Markov Chain Monte Carlo), making both approaches highly scalable. In terms of features, Prob-EC is currently a subset of PEC+, but the generalisation of the law of inertia provided by MLN-EC is a step ahead both PEC+ and EPEC. However, an advantage of EPEC over both these frameworks is that it supports imperfect sensing actions and conditional events.

Chapter 8

Discussion

This chapter provides a summary of the main contributions of this work, and suggests some possible directions for further research.

8.1 Contributions

This thesis contributes to the field of Artificial Intelligence. These contributions are mainly in the areas of Reasoning About Actions, Knowledge Representation, Probabilistic and Epistemic reasoning. Two new action languages are introduced, PEC+ and EPEC, able to reason about an agent's degree of belief.

While most work in the area of Reasoning About Actions is focused on planning and based on the Situation Calculus (see section 2.1.2), the main aim of this thesis is to develop a *narrative* formalism based on the Event Calculus (see section 2.1.3). PEC+ and EPEC contribute to this field by integrating and extending epistemic, probabilistic and narrative reasoning. For example, EPEC's representation of conditional actions (with its associated fixpoint semantics) is an advance over previous work, as it is able to model actions that can be conditioned on the strength of belief in a given property of the domain, even when the agent can only sense imperfectly. This is particularly appropriate for planning under high levels of uncertainty (see e.g. the medical scenarios described in section 1.2), where perfect sensing is not a realistic approximation. Within the set of EC formalisms, PEC+ and EPEC have pushed forward the integration of the original EC with probabilistic reasoning by generalising some typical EC features to the probabilistic case, for instance triggered actions. It is also worth mentioning that EPEC is the only probabilistic EC-based formalism to date which supports epistemic reasoning. This implies that its focus and potential areas of application differ from previous probabilistic EC languages which, as noted in section 7.2, are mostly used as event recognition tools.

This thesis also shows how ASP, a logic programming language, can be used to implement PEC+ using only a minimal amount of pre-processing and post-processing. The correctness of this implementation is then also proved. The thesis shows empirically (see section 4.4) that this implementation is not suitable for some domains (for example those with triggered actions) where the number of worlds grow in such a way as to make automated reasoning intractable. However, the implementation scales adequately for domains where various features such as the number of actions performed do not cause an explosive growth of the number of worlds.

To overcome computational difficulties in other cases, an alternative implementation of PEC+ is also provided, which uses a probabilistic programming language to perform approximate reasoning. Chapter 6 demonstrates that this implementation provides an approximation of the correct results, as it is validated against the ASP implementation where tractable, and that it also scales well for domains where a correct computation of the results is intractable.

This thesis also makes a contribution to the field of *Explainable AI*[1] in the sense that it provides frameworks for transparent reasoning in uncertain time-dependent domains. The validity of reasoning based on PEC+ or EPEC rests upon the validity of three components: the domain description, the domain independent assumptions (e.g., Closed World Assumption, Persistence and Justified Change) and the axioms of Probability Theory. The logical nature of these components makes the reasoning easily explainable in human terms, and in particular Probability Theory provides a rational mechanism for explaining degrees of uncertainty.

To summarise, this thesis contributes to the field of Artificial Intelligence by introducing a novel *combination* of

- Reasoning About Actions
- Narrative Reasoning
- Uncertain Reasoning
- Epistemic Reasoning
- Support for Belief-Conditioned Actions

in the form of two new action languages, namely:

- **PEC+**, which can represent a mix of uncertain information about specific event occurrences and initial conditions, triggered actions, and uncertain information about general causal rules.
- **EPEC**, which constitutes the main contribution of this thesis, builds upon PEC+ and can reason about future belief conditioned actions and their consequences in domains with imperfect sensing.

whose syntax and semantics constitute the *formal* contributions of this thesis. On the other hand, the *computational* contributions are two implementations of PEC+, namely:

- A provably correct ASP implementation
- A scalable, approximate implementation in Anglican

[1] In a nutshell, Explainable AI is a branch of Artificial Intelligence that is able to explain and justify its actions in a human-understandable way. As pointed out in section 2.2.3, there are strong arguments in support of the use of probabilistic belief functions (e.g., an agent using a probabilistic belief function is "rational" in the sense that it cannot be Dutch Booked and satisfies Cox's axioms). This is in contrast to other types of Artificial Intelligence which employ complex, non-transparent algorithms that make it hard to explain their decisions even for their developers.

8.2 Further work

This thesis constitutes a further step towards the establishment of a probabilistic epistemic Event Calculus, able to reason about a comprehensive spectrum of complex domains. There are many ways in which this work could be fruitfully continued. Some possible extensions and variations are described below.

If needed, PEC+ and EPEC allow for non-probabilistic extensions which could be employed to capture heuristic ways of reasoning. For instance, [64] shows how fuzzy sets theory can be applied in the context of narrative modelling and understanding. Employing such non-probabilistic belief functions could then enable PEC+ and EPEC to model aspects of common sense reasoning that are not captured by probabilistic reasoning alone. However, this would somewhat weaken the degree up to which PEC+ and EPEC computations are explainable and justifiable, as non-probabilistic models do not always come with a justification in terms of rationality.

Some of the formalisms presented in chapter 7 have been extended with features that PEC+ and EPEC do not support. Some effort could be spent on implementing these ideas in PEC+ and EPEC. Among these features, supporting continuous probability distributions and handling ramifications would be an interesting next step.

The implementations of PEC+ and EPEC provided in this thesis focus on temporal projection. It would be interesting to embed these implementations within an abductive framework, for example to tackle tasks such as planning and explanation generation in uncertain domains. Algorithms could be developed, for example, to find (sub-)optimal plans to achieve a given goal with (near-)maximum probability.

Much of the work described in the thesis relies on a precise specification of the domain being fully specified a priori. As it is a probabilistic formalism, it is feasible to learn the parameters of the underlying probabilistic model from observations believed to derive from the same domain, e.g. in the context of the tuberculosis example medical records of treatments of past individuals. A related challenge is to discover the structure of a domain itself, e.g. by learning c-propositions outcomes, or adding new propositions, or enriching the underlying language of actions and fluents. Again this would require observations relating to the same domain.

Since PEC+ and EPEC are not tied to any specific programming language, they could be re-implemented in other languages to exploit their specific features (especially to overcome computational complexity issues). For example, the ASP implementation provided in this thesis could be rewritten in LP^{MLN} [39], a recent generalisation of the stable models semantics to the probabilistic case based on Markov Logic Networks.

In both the Situation Calculus and the Event Calculus there is a long tradition of using classical logic to extend the theories. It should be possible to express the semantics of both PEC+ and EPEC in classical logic, and this might be advantageous as regards exploring its formal equivalences to other frameworks under certain conditions. The ASP implementation of chapter 4 provides a good foundation for this idea.

Finally, the integration between logic and probability within Reasoning about Actions gives a new perspective on long standing issues such as the ramification and qualification problems. While the ramification problem is already tackled in other probabilistic work (see e.g. PAL and $\mathcal{E}+$ from chapter 7), the qualification problem has not yet been explored in this context, although

it has been studied in non-probabilistic EC e.g. in [37], where it is discussed in the context of elaboration tolerance. Non-probabilistic EC often copes with domains that are underspecified e.g. in terms of the initial state of the world using non-determinism. But in the probabilistic case it is not clear what to infer from domains that e.g. do not specify a probability distribution over initial states of the world (in EPEC terms, do not have an i-proposition). Exploring such domains constitutes an interesting challenge, and incorporating corresponding solutions in PEC+ and EPEC would arguably be a step forward in reflecting aspects of human reasoning.

Bibliography

[1] Laurent Abel, Jamila El-Baghdadi, Ahmed Aziz Bousfiha, Jean-Laurent Casanova, and Erwin Schurr. Human genetics of tuberculosis: a long and winding road. *Phil. Trans. R. Soc. B*, 369(1645), 2014.

[2] Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger. *The AWK Programming Language*. Addison-Wesley Longman Publishing Co., Inc., 1987.

[3] Fahiem Bacchus, Joseph Y. Halpern, and Hector J. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence*, 111(1-2):171–208, 1999.

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

[5] Vaishak Belle, Gerhard Lakemeyer, and Hector J. Levesque. A first-order logic of probability and only knowing in unbounded domains. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 893–899, 2016.

[6] Vaishak Belle and Hector J. Levesque. Reasoning about continuous uncertainty in the situation calculus. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 732–738. AAAI Press, 2013.

[7] Vaishak Belle and Hector J Levesque. A logical theory of localization. *Studia Logica*, 104(4):741–772, 2016.

[8] Ivan Bratko. *Prolog programming for artificial intelligence*. International Computer Science Series. Pearson Education Canada, 4ed. edition, 2012.

[9] Keith L. Clark. Negation as failure. In Jack Minker, editor, *Logic and Data Bases*, volume 1, pages 293–322. Plenum Press, New York, London, 1978.

[10] Richard Threlkeld Cox. Probability, Frequency and Reasonable Expectation. *American Journal of Physics*, 14(1):1–13, 1946.

[11] Fabio Aurelio D'Asaro, Antonis Bikakis, Luke Dickens, and Rob Miller. Foundations for a probabilistic event calculus. In Marcello Balduccini and Tomi Janhunen, editors, *Logic Programming and Nonmonotonic Reasoning*, pages 57–63, Cham, 2017. Springer International Publishing.

[12] Bruno de Finetti. Sul significato soggettivo della probabilità. *«Fundamenta Mathematicae*, 17:298–329, 1931.

[13] Bruno De Finetti. *Philosophical Lectures on Probability: collected, edited, and annotated by Alberto Mura*, volume 340. Springer Science & Business Media, 2008.

[14] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI*, volume 7, pages 2462–2467, 2007.

[15] Daniel C. Dennett. Cognitive wheels: The frame problem of ai. In C. Hookway, editor, *Minds, Machines and Evolution*. Cambridge University Press, 1984.

[16] Murthy Devarakonda, Dongyang Zhang, Ching-Huei Tsou, and Mihaela Bornea. Problem-oriented patient record summary: An early report on a watson application. In *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*, pages 281–286, Oct 2014.

[17] David M. Eddy. Probabilistic reasoning in clinical medicine: Problems and opportunities. In Daniel Kahneman, Paul Slovic, and Amos Tversky, editors, *Judgment Under Uncertainty: Heuristics and Biases*, pages 249–267. Cambridge University Press, 1982.

[18] Thomas Eiter and Thomas Lukasiewicz. Probabilistic reasoning about actions in non-monotonic causal theories. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, UAI'03, pages 192–199, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[19] Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. *Recursive aggregates in disjunctive logic programs: Semantics and complexity*. Springer, 2004.

[20] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3), 2010.

[21] David Ferrucci, Anthony Levas, Sugato Bagchi, David Gondek, and Erik T. Mueller. Watson: Beyond jeopardy! *Artificial Intelligence*, pages 93–105, 2013.

[22] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1972.

[23] Michael Fogus and Chris Houser. *The joy of Clojure*.

[24] James Garson. Modal logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer 2014 edition, 2014.

[25] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Clingo = asp + control: Preliminary report. volume arXiv:1405.3694v1.

[26] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. Potassco: The potsdam answer set solving collection.

[27] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. pages 1070–1080. MIT Press, 1988.

[28] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *The Journal of Logic Programming*, 17(2):301–321, 1993.

[29] Kahl Y. Gelfond M. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. CUP, draft edition, 2014.

[30] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153:49–104, 2004. Logical Formalizations and Commonsense Reasoning.

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

[32] Douglas R. Hofstadter. *Godel, Escher, Bach: An Eternal Golden Braid*. Basic Books, Inc., New York, NY, USA, 1979.

[33] Luca Iocchi, Thomas Lukasiewicz, Daniele Nardi, and Riccardo Rosati. Reasoning about actions with sensing under qualitative and probabilistic uncertainty. 10(1), 2009.

[34] Edwin Thompson Jaynes and G. Larry Bretthorst, editors. *Probability theory : the logic of science*. Cambridge University Press, Cambridge, UK, New York, 2003.

[35] Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

[36] Antonios Kakas and Rob Miller. A simple declarative language for describing narratives with actions. *The Journal of Logic Programming*, 31(1–3):157–200, 1997. Reasoning about Action and Change.

[37] Antonis Kakas, Loizos Michael, and Rob Miller. Modular-E and the role of elaboration tolerance in solving the qualification problem. *Artificial Intelligence*, 175(1):49–78, 2011.

[38] Robert Kowalski and Marek Sergot. A logic-based calculus of events. In *Foundations of knowledge base management*, pages 23–55. Springer, 1989.

[39] Joohyung Lee, Yunsong Meng, and Yi Wang. Markov logic style weighted rules under the stable model semantics. In *CEUR Workshop Proceedings*, volume 1433, 01 2015.

[40] John Levine and Levine John. *Flex & Bison*. O'Reilly Media, Inc., 1st edition, 2009.

[41] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In *Proceedings of the Eleventh International Conference on Logic Programming*, pages 23–37, Cambridge, MA, USA, 1994. MIT Press.

[42] Jiefei Ma, Rob Miller, Leora Morgenstern, and Theodore Patkos. An epistemic event calculus for asp-based reasoning about knowledge of the past, present and future. In *Proc. of the 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR-19*, 2013.

[43] Paulo Mateus, Antonio Pacheco, Javier Pinto, Amilcar Sernadas, and Cristina Sernadas. Probabilistic situation calculus. *Annals of Mathematics and Artificial Intelligence*, 32(1/4):393–431, January 2001.

[44] John McCarthy. Programs with common sense. *Semantic Information Processing*, 1:403–418, 1959.

[45] John McCarthy. Situations, actions, and causal laws. Technical report, DTIC Document, 1963.

[46] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1–2):27–39, 1980. Special Issue on Non-Monotonic Logic.

[47] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[48] Rob Miller and Murray Shanahan. Narratives in the situation calculus. *Journal of Logic and Computation*, 4(5):513–530, 1994.

[49] Rob Miller and Murray Shanahan. Some alternative formulations of the event calculus. In *Computational logic: logic programming and beyond*, pages 452–490. Springer, 2002.

[50] Robert C. Moore. A Formal Theory of Knowledge and Action. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 319–358. Ablex, Norwood, NJ., 1985.

[51] Jeff B. Paris. *The uncertain reasoner's companion: a mathematical perspective*, volume 39. Cambridge University Press, 2006.

[52] Edwin P. D. Pednault. Formulating multiagent, dynamic-world problems in the classical planning framework. In M. P. Georgeff and A. L. Lansky, editors, *Reasoning about Actions and Plans*, pages 47–82. Kaufmann, Los Altos, CA, 1987.

[53] Frank P. Ramsey. Truth and probability. In R. B. Braithwaite, editor, *The Foundations of Mathematics and other Logical Essays*, chapter 7, pages 156–198. McMaster University Archive for the History of Economic Thought, 1926.

[54] Raymond Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy*, 27:359–380, 1991.

[55] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, Massachusetts, MA, illustrated edition edition, 2001.

[56] Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In *In Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 689–695. AAAI Press/The MIT Press, 1993.

[57] Richard B. Scherl and Hector J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1–2):1–39, 2003.

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

[59] Murray Shanahan. The frame problem. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2016 edition, 2016.

[60] Patrik Simons, Ilkka Niemelá, and Timo Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, June 2002.

[61] Anastasios Skarlatidis, Alexander Artikis, Jason Filippou, and Georgios Paliouras. A probabilistic logic programming event calculus. *Theory and Practice of Logic Programming*, 15:213–245, 3 2015.

[62] Anastasios Skarlatidis, Georgios Paliouras, Alexander Artikis, and George A Vouros. Probabilistic event calculus for event recognition. *ACM Transactions on Computational Logic (TOCL)*, 16(2):11, 2015.

[63] David Tolpin, Jan Willem van de Meent, Hongseok Yang, and Frank Wood. Design and implementation of probabilistic programming language anglican. *arXiv preprint arXiv:1608.05263*, 2016.

[64] Enric Trillas and Sergio Guadarrama. Fuzzy representations need a careful design. *International Journal of General Systems*, 39(3):329–346, 2010.

[65] Alan Mathison Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[66] Michiel Van Lambalgen and Fritz Hamm. *The proper treatment of events*, volume 6. John Wiley & Sons, 2008.