Università degli Studi di Milano

Dipartimento di Filosofia "Piero Martinetti"
Scuola di Studi Umanistici
Corso di Dottorato di Ricerca in Filosofia e Scienze dell'Uomo

PROBABILISTIC MODEL CHECKING WITH MARKOV MODELS SEMANTICS

New Developments and Applications

Tesi di Dottorato

Candidato:
Dr. Alberto Termine

Supervisore:
Prof. Giuseppe Primiero

Ciclo di Dottorato XXXV

Abstract

Contemporary society is increasingly dependent on the use of autonomous computational systems. Being sure that these systems behave appropriately and do what they have been designed for is thus fundamental. Several techniques have been developed to this purpose. Among them, *model checking* includes a series of methods that make use of computational logic and automatic procedures to check whether systems' behaviour satisfies given desirable properties.

In recent years, due to the increasing prevalence of stochastic models, conventional model checking techniques have proven inadequate. The consequence has been the development of *probabilistic model checking*, a new research program aimed at developing proper tools to check the behaviour of stochastic systems. Among others, the probabilistic model checking with Markov models semantics and its related languages (i.e., the Probabilistic Computation Tree Logic (PCTL) and its extensions) has been particularly successful and is nowadays applied throughout a variety of different systems and domains. Nevertheless, there remain numerous unexplored developments and applications in this field. In this doctoral dissertation, we specifically concentrate on three such aspects that hold significant relevance to contemporary AI research. To each of these aspect, we will dedicate a chapter. The dissertation is structured as follows.

In the first chapter, we present the essential foundational information regarding probability theory and Markov models. Initially, we introduce fundamental terminology and refresh the reader's memory on the significant philosophical interpretations of probability, the Kolmogorov axiomatization, and the associated calculus. Subsequently, we delve into the application of probability in studying stochastic systems. Our primary emphasis is on Markov models, including discrete-time Markov chains, Markov reward models, and Markov decision processes. These models are indeed considered the reference formalism for representing stochastic systems in the domain of model checking.

In the second chapter, we provide an overview of the state of the art of model checking, with a particular emphasis on both single-agent and multi-agents probabilistic model checking. In the opening section, we discuss the fundamental ideas of the *program verification* research field, of which model checking represents one of the contemporary evolutions.

We focus in particular on the philosophical debate that emerged in the 1980s between

advocates and opponents of formal methods in computer science. This debate can be considered a pivotal milestone in the shift from traditional program verification methods to modern techniques such as model checking.

In the subsequent sections, we then introduce various formalisms utilized in model checking for representing computational systems, specifying their attributes, and verifying their behaviors. For standard (non probabilistic) model checking, these formalisms include transition systems and their related logical languages (i.e., LTL and CTL), while for probabilistic model checking they include Markov models and the related language PCTL. We conclude the chapter by examining some recent extensions of probabilistic model checking suitable for the analysis of epistemic and probabilistic properties of stochastic multi-agent systems. Notably, the formalisms we consider include the logics CTLK [113], PCTLK [175], and COGWED [28], where the latter two formalisms will serve as the foundation for subsequent advancements and applications discussed in the remaining chapters of the dissertation.

In the third chapter, we explore a potential connection between probabilistic model-checking and eXplainable AI (XAI), a recently-born field of research aimed at developing methods and tools to make opaque machine learning systems more humanly understandable. The chapter is divided in two main parts. In the first part, we analyse the so-called *opacity problem* from a more philosophical point of view, notably focusing on elucidating its many dimensions and explaining how these are actually addressed within the XAI research program. In the second part, we restrict our focus to a specific XAI framework, i.e., *post-hoc explanation methods based on surrogate models*. More specifically, we propose a logic and related model-checking procedures for specifying and checking relevant reliability properties of XAI-explanations for opaque ML systems provided via surrogate models. Among relevant explanations reliability properties, we consider in particular *transparency*,*accuracy*, and *trustworthiness*. We introduce a language with appropriate operators to specify such properties with respect to a single or a group of surrogate models. We propose a semantics for these operators based on a multi-agent Markov structure and develop related model-checking algorithms based on extending established existing procedures introduced in Chapter 2. Finally, we conclude with some computational complexity considerations and remarks about further developments of this research.

In the fourth chapter, we present an extension of probabilistic model checking to the domain of imprecise probabilities. We start with a brief introduction of the latter, focusing in particular on the reasons motivating its usefulness in the model checking domain. We argue that imprecise probabilistic models may be of fundamental importance for model checking applications, particularly since they permit to relax the stationarity assumptions associated with the utilization of Markov models. This allows for the analysis of non-stationary systems (i.e., systems whose stochastic behavior changes over time) and systems whose behaviour is not fully known. In the first part of the chapter, we introduce

imprecise Markov models for both single and multi-agent systems and present feasible methods to compute relevant inferences over them. In the second part, we present a logic that enables the specification of properties for systems represented by imprecise Markov model. The logic is obtained as an extension of the well-known PCTL and it is called *Epistemic-Imprecise Probabilistic Computation Tree Logic* (EIPCTL). We provide this logic with a suitable semantics defined over imprecise Markov models and related model checking procedures based on feasible inferences discussed in the first part of the chapter. We conclude the chapter by outlining some computational complexity considerations. Notably, we argue that one of the greatest advantages of working with imprecise probabilities is the possibility to relax stationarity assumptions, which limit the applicability of model checking, without incurring in an increase of computational complexity. In fact, we demonstrate that all the relevant tasks involved in model checking, which can be resolved in standard models by solving linear equation systems, can be resolved in imprecise models by solving linear programming tasks whose complexity is always polynomial in the models' number of states.

In the fifth chapter, we present a novel application of Markov model semantics to the problem of logical omniscience, which has been considered one of the major issues in the development of an epistemic logic suitable for representing real-world agents since the 1960s. In the first part of the chapter, after a brief introduction to epistemic logic, we analyze a variety of solutions proposed for the problem of logical omniscience, both inside and outside the model-theoretic tradition. For each of these solutions, we briefly analyze the reasons that make them not completely satisfactory. In the second part, we introduce our new framework for epistemic logic, which is based on understanding real-world agents' deductive reasoning in terms of a "state-space exploration task" that can be modeled via Markov decision processes and reinforcement learning. Within this framework, we provide a new definition of explicit knowledge according to which an agent knows explicitly a certain proposition if and only if the agent is able to derive that proposition from the information it is actually aware of by making the best use of its finite resources. Based on this definition, we introduce a new logic of knowledge and awareness called "Markov Deduction Structures Logic." By exploiting a combination of probabilistic model checking and reinforcement learning techniques, we then outline a semantics and relative model checking framework for this logic. In the third part, we present a dynamic multi-agent extension of the logic for modeling agents' actions. These actions may include learning inferential rules from other agents or exchanging resources and information with them. Although we do not offer any experimental evaluation of our framework, we advance some considerations concerning its implementation in the conclusions.

The dissertation builds upon a range of preliminary results previously presented at multiple international workshops and conferences, and subsequently published in scientific

journals and conference proceedings.

The opacity taxonomy included in chapter 3 is largely based on the analogous taxonomy developed in [62]. The Ex-PCTL logic presented also in chapter 3 resumes and extends the ATCTL (Accuracy and Trustworthiness Computation Tree Logic) developed in [160]. The main results in Chapter 4 have been collected in a journal paper forthcoming in SN computer science. They unify and extend two previous works on model checking with imprecise Markov models: [157] and [158]. The former develops a model checking framework for *imprecise Markov reward models*, while the latter focuses on model-checking stochastic multi-agent systems with imprecise probabilities. An early draft of the framework developed in chapter 5 was presented at the Logic Colloquium 2021, while the extended version here presented has been recently submitted to IJAR.

Riassunto

La società contemporanea è sempre più dipendente dall'uso di sistemi computazionali autonomi. Essere sicuri che questi sistemi svolgano in modo appropriato la funzione per cui sono stati progettati risulta pertanto di fondamentale importanza. Il *model checking* rientra tra le diverse metodologie che sono state sviluppate a questo scopo. Esso si caratterizza come un eterogeneo campo di ricerca che include differenti procedure automatiche per verificare che i comportamenti dei nostri sistemi computazionali soddisfino determinate proprietà desiderabili, queste ultime in genere specificate sotto forma di formule logiche in un dato linguaggio formale.

Negli ultimi anni, le tradizionali tecniche di model-checking stanno diventando via via sempre piú inadeguate perlopiú a seguito dell'emergere del *machine learning* e di una nuova generazione di sistemi di IA stocastici che su di esso sono basati. In risposta a tale inadeguatezza sono stati inaugurati diversi programmi di ricerca dedicati alla verifica di sistemi stocastici. Tra quelli di particolare successo troviamo il model checking probabilistico basato su semantiche Markoviane, il quale è oggigiorno applicato ad un'ampia varietà di domini. Vi sono tuttavia molti sviluppi e applicazioni di quest'ultimo che non sono stati ancora presi in considerazione. In questa tesi di dottorato, ci concentriamo su tre di essi che risultano particolarmente rilevanti per gli attuali sviluppi della ricerca in IA. A ciascuno di essi dedichiamo un capitolo della tesi, secondo la struttura di seguito delineata.

Il primo capitolo fornisce le necessarie conoscenze di base inerenti la teoria della probabilità e i modelli di Markov. Si suddivide in due parti. La prima introduce la terminologia di base, passando in rassegna le più rilevanti interpretazioni filosofiche della probabilità e presentando la ben nota assiomatizzazione di Kolmogoroff. La seconda parte spiega come applicare la probabilità allo studio dei sistemi stocastici. Si concentra in particolare sui cosiddetti *modelli di Markov* (catene di Markov a tempo discreto, catene Markov con rewards, e processi decisionali di Markov) che rappresentano il formalismo di riferimento per la modellizzazione dei sistemi stocastici nell'ambito del model checking probabilistico.

Il secondo capitolo fornisce una panoramica dello stato dell'arte nell'ambito del model checking. La sezione iniziale presenta brevemente l'ambito di ricerca noto come *program verification*, di cui il model checking rappresenta una delle piú recenti evoluzioni. Essa si

concentra in particolare sul dibattito filosofico degli anni '80 tra sostenitori e critici dell'uso dei metodi formali nell'informatica, dibattio che riveste un'importanza fondamentale nel passaggio dalle vecchie alle più recenti ed avanzate tecniche di program verification come il model checking. Nelle sezioni successive, vengono presenti diversi formalismi utilizzati nell'ambito del model checking per l'analisi di differenti categorie di sistemi computazionali e delle loro proprietá. Per il model checking standard (non probabilistico), questi formalismi includono i sistemi di transizione e i relativi linguaggi logici (ad esempio, LTL e CTL), mentre per il model checking probabilistico essi includono i modelli di Markov e la logica PCTL. Il capitolo si conclude con l'esame di alcune recenti estensioni del model checking probabilistico specificatamente concepite per l'analisi di sistemi stocastici multi-agente. Tra questi si sofferma in particolare sulle logiche CTLK [113], PCTLK [175] e COGWED [28],le quali rappresentano il punto di partenza per gli sviluppi e le applicazioni che vengono presentati nei restanti tre capitoli della tesi.

Il terzo capitol, esplora la potenziale connessione tra il model-checking probabilistico e l'*eXplainable AI* (XAI), un campo di ricerca nato di recente e volto a sviluppare metodi e strumenti per rendere i sistemi di machine learning più comprensibili agli utenti. Il capitolo è diviso in due parti principali. La prima parte del capitolo é dedicata all'analisi filosofica del cosiddetto *problema dell'opacità* e alla spiegazione di come questo venga affrontato nell'ambito dell'Explainable AI. La seconda parte del capitolo si focalizza su una specifica classe di metodi XAI, ovvero i *metodi di post-hoc explanation basati su modelli surrogati*. Nello specifico, il capitolo introduce una logica per la specifica di alcune proprietá desiderabili dei modelli surrogati quali: la *trasparenza*, l'*accuratezza* e l'*affidabilità*. Viene quindi sviluppata una semantica per questa logica basata su una classe di strutture multi-agente e sull'estensione delle procedure di model checking introdotte nel Capitolo 2. Il capitolo si conclude con alcuni esempi di applicazioni del formalismo sviluppato.

Nel quarto capitolo, viene discussa una possibile estensione del model checking probabilistico nell'ambito delle *probabilità imprecise*. Le prime sezioni del capitolo forniscono una breve introduzione a questo formalismo e alle ragioni che ne motivano l'utilità nel dominio del model checking. Nello specifico, viene sottolineato come le versioni "imprecise" dei modelli di Markov presentati nel capitolo 2 possano giocare un ruolo fondamentale per le applicazioni del model checking probabilistico nella misura in cui consentono di rilassare le ipotesi di stazionarietà richieste dai modelli di Markov. In questo modo essi rendono possibile tanto l'analisi di sistemi stocastici non-stazionari (ovvero, sistemi il cui comportamento stocastico cambia nel tempo), quanto quella di sistemi il cui comportamento stocastico non puó essere determinato con esattezza. La seconda parte del capitolo introduce una nuova logica per l'analisi di sistemi stocastici multi-agente chiamata *Epistemic-Imprecise Probabilistic Computation Tree Logic*. Venogno quindi introdotte una semantica per questa logica, basata su modelli di Markov imprecisi, e relative procedure di model checking. Il capitolo si conclude con alcune considerazioni inerenti la complessità com-

putazionale dei modelli di Markov imprecisi. Viene sottolienato nello specifico come uno dei maggiori vantaggi di lavorare con le probabilità imprecise sia la possibilità di allentare le ipotesi di stazionarietà, che notoriamente limitano l'applicabilità del model checking probabilistico, senza per questo incorrere in un aumento della complessità computazionale complessiva degli algoritmi utilizzati.

Il quinto capitolo presenta una nuova applicazione della semantica dei modelli di Markov al ben noto problema dell'*onniscienza logica*, considerato sin dagli anni '60 come una delle problematiche principali che si frappongono allo sviluppo di una logica epistemica adatta alla rappresentazione di agenti reali. Dopo una breve introduzione al formalismo della logica epistemica, vengono presentate alcune delle piú note soluzioni al problema dell'onniscienza logica sviluppate negli anni e discusse brevemente le ragioni che le rendono non del tutto soddisfacenti. La seconda parte del capitolo introduce un nuovo framework per la logica epistemica basato sul rappresentare il ragionamento deduttivo come un *compito di esplorazione spaziale* modellizzabile tramite processi decisionali di Markov e *reinforcement learning*. Viene quindi sviluppata una nuova definizione di *conoscenza esplicita* secondo cui *un agente conosce esplicitamente una certa proposizione se e solo se esso è in grado di derivare tale proposizione dalle informazioni di cui é attualmente consapevole (aware) ottimizzando al meglio le sue risorse finite*. Sulla base di questa definizione, viene introdotta una nuova logica della conoscenza e della consapevolezza chiamata *Markov Deduction Structures Logic*. La semantica e il relativo framework di model checking per questa logica vengono sviluppati sfruttando una combinazione di tecniche prese a prestito dal reinforcement learning. La terza parte del capitolo sviluppa un'estensione dinamica della logica precedentemente introdotta in grado di modellizzare diverse tipologie di azioni degli agenti, quali l'azione di *apprendere* nuove regole d'inferenza da altri agenti o l'azione di *scambiare* con essi le proprie risorse e informazioni. Nonostante non venga fornita alcuna valutazione sperimentale del framework, alcune considerazioni riguardanti la sua implementazione vengono delineate nelle conclusioni.

La tesi si basa su di una serie di risultati preliminari presentati, e in parte pubblicati, in alcuni workshop e conferenze internazionali negli ultimi anni. La tassonomia dell'opacità inclusa nel capitolo 3 è largamente basata sull'analoga tassonomia sviluppata in [62]. La logica Ex-PCTL, anch'essa presentata nel capitolo 3, riprende ed estende la logica ATCTL (Accuracy and Trustworthiness Computation Tree Logic) sviluppata in [160]. Il capitolo 4 unifica ed estende due lavori precedenti inerenti il model checking con modelli di Markov imprecisi, ovvero [157] e [158]. Il primo di questi sviluppa una serie di tecniche di model checking per la controparte imprecisa delle catene di Markov con reward, mentre il secondo introduce un framework di model checking per sistemi stocastici multi-agente basato su una nuova classe di modelli chiamati *Imprecise Probabilistic Interpreted Systems*. Una prima bozza del framework sviluppato nel capitolo 5, infine, è stata presentata al Logic Colloquium 2021.

Contents

Chapter 1

Probability and Markov Models

Abstract

Probability theory plays a central role in the formal verification of computational systems that exhibit stochastic behaviors, which are the focus of the present dissertation. This chapter provides a general introduction to the theory of probability and its applications to stochastic systems. The chapter is divided into two parts. The first part introduces the fundamental terminology of probability, explores the major philosophical interpretations of this concept, and presents its standard axiomatization. The second part focuses on the application of probability to the specification and analysis of a specific class of stochastic systems that are of primary importance in the field of formal verification, namely *Markov models*.

1.1 Basic Terminology

In technical jargon, phenomena whose outcome cannot be predicted deterministically are called *aleatory experiments*. Given an aleatory experiment, the set of all its possible outcomes is called the *sample space*. This is usually denoted by Ω, while notation $\omega \in \Omega$ is used to denote the elements of Ω that correspond to the various possible elementary outcomes of the experiment. For example, the aleatory experiment "toss of a coin" is characterized by a sample space $\Omega := \{\text{heads}, \text{tails}\}$ that includes two possible elementary outcomes: "heads" and "tails". In general, there are no constraints on the nature of Ω, which can consist of either a finite, a countable or a continuous space.

When we reason about an aleatory experiment, we might be interested either in the probability of an elementary outcome or in that of a more complex one. For example, we might be interested in the probability that the outcome of a die roll is 1 (elementary outcome), but also in the probability that the outcome is an odd number (i.e., it is either 1, 3, or 5). In the latter case, the probability we want to calculate is not of a single element $\omega \in \Omega$ but of a specific subset of the sample space $B \subseteq \Omega$. We call this an *event* and we call

probability of an event $B \subseteq \Omega$ the probability that the outcome of the experiment is any random $\omega \in B$. When an event B corresponds to $\emptyset$, we refer to it as the *impossible event*. In the toss of a coin, the event "the outcome is neither heads nor tails" is an example of impossible event. Differently, when an event B equals the whole Ω, we refer to it as the *certain event*. In the roll of a die, "the outcome is a natural number between one and six" is an example of certain event. Finally, if two events B and D have empty intersection, i.e., if $B \cap D = \emptyset$, we say that B and D are *mutually exclusive* (or *disjoint*) events.

Events form a proper family of sets called the *family of events*. To avoid well-known paradoxes that arise when Ω is a non-measurable space, it is common to assume the family of events to coincide with the σ-algebra of the subsets of Ω, here denoted $\sigma(\Omega)$ [142]. The latter is defined as the set of all proper subsets of Ω that satisfies the following conditions:

- $\Omega \in \sigma(\Omega)$,
- If an event is included in Ω, then it is its complement, i.e., $B \in \sigma(\Omega) \implies B^c \in \sigma(\Omega)$, where B^c denotes the complement of B,
- Given a succession of events $\{B_n\}_{n=1}^{\infty}$, if these events are all included in $\sigma(\Omega)$, then it is its union, i.e., $(\forall n \in \mathbb{N}) B_n \in \sigma(\Omega) \rightarrow \bigcup_{n=1}^{\infty} B_n \in \sigma(\Omega)$.

When $\Omega = \mathbb{R}$, the σ-algebra considered is usually the Borel's σ-algebra, i.e., the smallest σ-algebra that is generated by open sets of $\mathbb{R}$ through the operations of *countable union*, *countable intersection*, and *relative complement* [142]. In different cases, other σ-algebra can be considered [68, Ch.1].

1.2 The Philosophical Meaning of Probability

Philosophers have long debated the nature and meaning of probability, providing different ways of defining and interpreting it. Since the last century, three main interpretations have emerged in the debate: the *classical interpretation*, the *frequentist interpretation*, and the *subjectivist interpretation*. Alongside with the latter, other philosophical views of probability have been proposed, including, for example, Carnap's *logicist interpretation* [25], and Popper's *theory of propensions* [130]. In this work, we provide only a brief overview of the major interpretations. The reader interested in a more in-depth overview of the various philosophical interpretations of probability can refer to [88, 152].

1.2.1 Classical Interpretation.

The roots of the classical interpretation of probability trace back to the famous 1654 correspondence between Blaise Pascal and Pierre de Fermat on the topic of *gambling*, which

is also closely related to the first developments of the mathematical theory of probability [162]. However, the first explicit and clear formulations of this interpretation appear only later in the works of Jacob Bernoulli [17] and in the *Theorie Analytique des Probabilites* of French mathematician Pierre Simon de Laplace [44]. The latter writes:

"The theory of chances consists in reducing all events of the same kind to a certain number of equally possible cases, that is to say, to cases whose existence we are equally uncertain of, and in determining the number of cases favourable to the event whose probability is sought. The ratio of this number to that of all possible cases is the measure of this probability, which is thus only a fraction whose numerator is the number of favourable cases, and whose denominator is the number of all possible cases[44, p.4]."

As emerges from Laplace's words, the probability of an event B to occur in the execution of an experiment E is here defined as the ratio between the number of outcomes of E favorable to the occurrence of B and the number of all possible outcomes of E. This definition works only under the assumption that:

1. the number of possible outcomes of E is always finite;
2. all possible outcomes of E are equally probable.

These assumptions severely limit the applicability of the classical interpretation. Consider for example the tossing of a rigged coin. Our intuition tells us that, since the coin is rigged, the probability of outcome "heads" is different from that of outcome "tails". If we apply the classical interpretation, however, we obtain that both outcomes "heads" and "tails" have the same probability to occur, which is clearly absurd. How can we define the probability of an event like "the outcome is heads" in such very common cases? the classic interpretation does not offer an answer. Furthermore, the classical interpretation requires that subjects know all the possible cases and are able to describe and enumerate them [152]. This requirement is also problematic and we can find many cases where it cannot be satisfied. Ultimately, the classical interpretation has too many limitations and is not applicable in a systematic way. This is the reason why scholars have been seeking for alternative and more effective interpretations since the nineteenth century. Among these, the frequentist and subjectivist interpretations have been, and continue to be, particularly successful.

1.2.2 Frequentist Interpretation.

The frequentist interpretation relies on the intuition that frequencies and probabilities are intimately related, so that we can identify the latter with the former. This interpretation

emerged at the end of the XIX century following the first applications of probability calculus to natural science and the development of statistical mechanics [152]. The simplest version of frequentism is called *finite frequentism* and was originally proposed by Venn [170, p.84]. It claims that: "the probability of an event B in a finite series A of subsequently executions of an experiment E is the relative frequency of actual occurrences of B within A". This interpretation is very closed to the classical one. In fact, it considers probability as the proportion of favorable outcomes on the number of total outcomes, where, in this case, the total outcomes are not the possible but the *actual* ones. Finite frequentism provides an operational definition of probability that may be considered suitable to statistical practice and is usually appreciated by strong empiricists [88]. Nevertheless, it runs into several paradoxes. The most famous is the *single-case paradox* that arises with scenarios in which an experiment is run only once. Suppose to toss a coin just once obtain the outcome "heads". According to finite frequentism, the probability of outcome "heads" in such a scenario should be equal to 1, while the probability of outcome "tails" should be 0. Clearly, this is absurd. A straightforward solution to this paradox may consist of asking the series A to be sufficiently long. Unfortunately, this solution is not very effective, as it makes the notion of probability vague and not rigorous. After all, it is not clear what counts as a "sufficiently" long series, as well as it is totally implausible that an agent is always able to determine when the number of executions of an experiment is sufficiently high to return a realistic probability value. A possible way out from this paradox, proposed by Reichenbach [137] and Von Mises [172], consists of considering infinitely long series and defining the probability of B as the limit to which the relative frequency of occurrences of B tends as the number n of executions of E increases infinitely, i.e., $P(B) := \lim_{n\to\infty} \frac{n_B}{n}$, where n_B denotes the number of executions with outcome B. This refined version of frequentism is usually called *hypothetical frequentism* as the notion of "infinite series" has no actual meaning (there exist no such infinite series in the actual world) but represents an hypothetical counterfactual condition, that is, the probability of B is equal to the relative frequency of B within A *under the counterfactual hypothesis* that A is infinitely long [88]. This interpretation overcomes the limits of finite frequentism but it is not free of issues. First of all, it introduces a counterfactual condition that violates the verification principle and makes the frequentist interpretation incompatible with radical forms of empiricism[1]. In fact, if probability statements describe counterfactual conditions, then, by definition, there exist no actual scenarios that allow to verify probability statements and, consequently, for the verificationist probability statements are meaningless and devoid of epistemic content. Moreover, there are issues stemming from the fact that many events naturally occur only once and are not repeatable. For example, the event: "election of Sergio Mattarella as president of the Italian republic in 2022". How can one

[1] I say "radical" forms because there exist weak forms of empiricism, adopted for example by Carnap [25] and Reichenbach [138] in their more mature works, which are compatible with this interpretation.

compute the probability of such events? the hypothetical frequentism is unable to provide an answer. A possible way out could be to argue that such events are not common in science and, thus, fall outside the specific interest of the frequentist interpretation, which is to clarify how the notion of probability should be used in scientific practice. However, experiments characterized by the non-replicability are becoming increasingly frequent also in scientific research [141] and place a serious limit on the applicability of the frequentist account. Nonetheless, this interpretation of probability continues to be widely adopted in statistics and experimental sciences, at least because, on the pragmatic side, it seems to offer a simple way to estimate the probabilities of events [152]. Before to conclude, we have to mention a criticism that is sometimes moved against frequentism and concerns the fact that this interpretation, being strictly connected to the idea of uncertainty as randomness, seems unable to account for those scenarios in which uncertainty is not due to randomness but is the by-product of epistemic ignorance. For more details on this and related critiques, we refer to [49] and [101, p.341-359].

1.2.3 Subjective Interpretation.

Opposed to the frequentist interpretation but equally diffused, especially in the domain of social sciences, is the subjectivist interpretation. This philosophical view of probability relies on the intuition that probabilities do not measure "real" properties of events but the degrees of belief, or confidence, of rational epistemic subjects in the occurrence of events. This intuition traces back to Augustus De Morgan, who writes: "By degree of probability, we really mean, or ought to mean, degree of belief." [46, p.1847]. In the twentieth century, this intuition was independently developed and made more rigorous by Frank Ramsey and Bruno De Finetti [43, 67, 134]. Both proposed to analyze probability in terms of betting behaviours of rational subjects. Let consider a lottery ticket that pays 1 euro[2] if the event B occurs and 0 otherwise, then ask a subject a how much she is willing to pay for the lottery ticket. If the subject is rational, she will answer an amount of dollars corresponding to a value p in the norm $[0, 1]$: p is the degree of belief of a in the occurrence of B, that is, the probability of B according to a[3]. The result of De Finetti and Ramsey's joint analyses is the so-called De Finetti-Ramsey Theorem[4] [152, p.16].

Theorem 1 (Ramsey-De Finetti) *Probabilities are degrees of belief on pain of coherence.*

[2]Specifically, De Finetti speaks in terms of *units of utility*, but for simplicity we adopt here an actual currency.

[3]For more details on Ramsey and De Finetti's works, see [152, p. 14-18].

[4]The latter is not properly a theorem but rather an *explication*, that is, a defining sentence that replaces the "informal" notion of *probability* with a concept that can be precisely and explicitly defined mathematically. For a detailed characterization of the notion of *explication*, see [107, sec.1.1].

As the reader may note, there is an additional requirement in the Theorem 1, namely *coherence*, which ensures that subjects satisfy rationality constraints. In particular, Ramsey [134] provides an analysis of rationality constraints in betting contexts formulated in terms of Dutch books, where "a Dutch book is a combination of bets that would make the agent lose utility no matter what happens in fact" [152, p.16]. Specifically, a subject is rational in a betting context if and only if she behaves in order to avoid Dutch books and, then, sure loss. Interestingly, it is possible to prove[5] that a subject in a betting context behaves to avoid Dutch books if and only if her beliefs conform to Kolomogorov's axioms of probability explained below in Section 1.3. This certainly represents a strong point in favor of subjectivism, which is gaining popularity not only among those who work on theoretical and foundational aspects of probability [152], but also in many empirically oriented parts of science, in particular as a grounding framework for Bayesian reasoning, nowadays considered the most valid alternative to frequentist statistics [85, 149]. Furthermore, as we will see in Chapter 2, the subjectivist interpretation is widely adopted also by proponents of the theory of imprecise probabilities. In particular, the most influential among foundational accounts for imprecise probabilities, namely the *Theory of Desirable Gambles* [173], is in fact a subjectivist account developed as a generalization of De Finetti's analysis of betting behaviours [174].

1.3 The Axiomatization of Probability

While the philosophical debate raged, the development of the mathematical theory of probability have been proceeding largely independently on its tracks. The turning point in the development of a complete and well-defined mathematical treatment of probability was the axiomatization proposed by the Russian mathematician Kolmogoroff at the beginning of the last century [94]. Following the spirit of Hilbert's program [178], Kolmogoroff introduced a set of axioms and basic calculation rules that provide an implicit definition of probability independent from and compatible with any philosophical interpretation of the concept. Since the years immediately following the release of Kolmogoroff's work, his axiomatization has become very famous and have been widely adopted among mathematicians, ending up with representing nowadays the hearth of the mathematical foundations of probability. In this section, we report Kolmogoroff's axiomatization and briefly describe its meaning.

[5] For the details of the proof, see [152, p.17-19].

1.3.1 Kolmogoroff's Axioms

Given a sample space Ω and the family of events $\sigma(\Omega)$ associated with it, we call *probability measure* any function $P : \sigma(\Omega) \mapsto [0, 1]$ that assigns to each event $B \in \sigma(\Omega)$ a real number $P(B)$ representing the *probability* of B. A measure $P : \sigma(\Omega) \mapsto [0, 1]$ defined over the family of events $\sigma(\Omega)$ is a *probability measure* if and only if, for all $B \in \sigma(\Omega)$, it satisfies the following axioms:

1. $0 \leq P(B) \leq 1$,
2. $P(\Omega) = 1$,
3. For any numerable sequence of mutually exclusive events $B_1, B_2, \ldots$: $P(\bigcup_{i=1}^{\infty} B_i) = \sum_{i=1}^{\infty} P(B_i)$

The first axiom establishes that a probability is always a real number in the norm $[0, 1]$. The second axioms captures the intuition that, in the execution of an experiment one of the outcome in the sample space Ω must occur and, consequently, the probability of Ω (i.e., the probability that at least one $\omega \in \Omega$ occurs) is always equal to 1. The third axiom capture the intuition that the probability of an event B_i, in a collection of mutually exclusive events $B_1, B_2, \ldots$, to occur is equal to the probability that either B_1 occurs, or B_2 occurs, *et cetera*. Actually, this is usually considered the least intuitive among the axioms, for more details on this point, see [152].

1.3.2 Operations with Probability

Starting from Kolmogorof's axioms, it is possible to derive some basic calculation rules that allow to perform inferences with probabilities. These rules, reported in Definition 1 allow for performing operations with probabilities and, together with axioms, form the so-called *probability calculus*.

Definition 1 (Rules of the probability calculus) *Let $\{E_i\}$ denote a total (finite or countable) partition of Ω and let B denote any possible event in $\sigma(\Omega)$. The following propositions express the rules of the probability calculus:*

$P(B) = 1 - P(B^c)$	rule of complementarity ;
$A \subseteq B \rightarrow P(A) \leq P(B)$	rule of implication ;
$P(A \cup B) = P(A) + P(B) - P(A \cap B)$	rule of general addittivity ;
$P(B) = \sum_i P(B \mid E_i) \cdot P(E_i)$	rule of total probability ;
$(\forall E_h \in \{E_i\})\ P(E_h \mid B) = \frac{P(B\|E_h)\cdot P(E_h)}{\sum_i [P(B\|E_i)\cdot P(E_i)]}$	Bayes′ rule .

1.4 Stochastic Variables

In general, studying complex aleatory phenomena based only on the sample space and the family of events is rather unfeasible. For example, consider the experiment "rolling of a fair dice" and imagine that you are interested in the event "the sum of the outcomes of the two dices is less than 20". In this scenario, the sample space Ω is given by all 10-tuples of possible outcomes of the two dice, i.e., $\Omega := \{\langle 1,1,6,2,4,3,5,6,2,2\rangle, \dots\}$ and the event "the sum of the outcomes of the two dice is less than 20" is the proper subset of Ω that includes all the 10-tuples $\langle x_1, \dots, x_n\rangle$ (with x_i outcome of the i-th dice) such that $\sum_{i=1}^{n} x_i \leq 20$. To deal with queries like "what is the probability that the sum of the outcomes of the two dice is less than 20?" is clearly impossible in this description format. Hence, we need a smarter way of representing and reasoning with probability. This can be obtained by defining a function $X : \Omega \rightarrow \mathbb{R}$ that associates each 10-tuple $\langle x_1, \dots, x_n\rangle$ (i.e., each $\omega \in \Omega$) with a real number $r \in \mathbb{R}$ that is the sum of all n elements of the tuple, i.e., $X(\omega) := \sum_{i=1}^{n} x_i$ for all $\omega \in \Omega$. The obtained format is easier to handle. For example, the query "what is the probability that the sum of the outcomes of the two dice is less than 20?" now can be written simply as "what is the probability that $X(\omega) \leq 20$?". The function X is an example of what in statistics is usually called a *stochastic variable*.

Definition 2 (Stochastic variable) *Given a probability space* $(\Omega, \sigma(\Omega), P)$*, we call* stochastic variable X *a correspondence between the elements* $\omega \in \Omega$ *and elements of a measurable space* $\mathbb{T}$ *such that, for all* $t \in \mathbb{T}$*:*

$$\{\omega \mid X(\omega) \leq t\} \in \sigma(\Omega)$$

That is, a stochastic variable is a correspondence between the sample space and the elements t of a measurable space $\mathbb{T}$ such that, for all $t \in \mathbb{T}$, the subset of Ω including all

outcomes $\omega : X(\omega) \leq t$ is a proper *event*[6]. This condition is fundamental as X represents the object to which we apply probability measures (our queries have the form "what is $P(X = x)$?") and, by definition, probability measures can be applied only to elements of a measurable space.

Stochastic variables can be either *categorical* or *continuous*. In general, a stochastic variable $X : \Omega \mapsto \mathbb{T}$ is called *categorical* if and only if $\mathbb{T}$ is a countable space; otherwise, X is called *continuous*. In this work, we almost always consider only categorical stochastic variables. Only these, indeed, are relevant to model *discrete-time* stochastic agents representing the target of our analyses in the next chapters.

Definition 3 (Probability distribution) *Given a stochastic variable X, a probability distribution over X, denoted by $P(X)$, is a function that assigns to each possible value x of X a number in the norm $[0, 1]$ such that, for all possible x, $P(X = x)$ denotes the probability that X takes value x.*

If X is categorical, then $P(X)$ is a (finite or infinitely countable) set that we can represent as discrete points on the space $P(X) \times \mathbb{T}$. Differently, if X is continuous, then $P(X)$ is an infinite set, that is, a seamless curve on the space $P(X) \times \mathbb{T}$.

Probability distributions can be either *simple*, *joint*, or *conditional*. A probability distribution $P(X)$ that concerns only one stochastic variable X is called *simple*. Given a collection of stochastic variables $X, Y, Z, S \ldots$, a *joint* probability distribution, denoted by $P(X, Y, Z, \ldots)$, is a probability distribution that assigns to each tuple $\langle x, y, z, \ldots \rangle$ the probability that events $X = x$, $Y = y$, $Z = z, \ldots$ jointly occur. On the other hand, a *conditional* probability distribution, denoted by $P(X \mid Y, Z \ldots)$, is a probability distribution that assigns to each tuple $\langle x, y, z, \ldots \rangle$ the probability that event $X = x$ occurs *given that* $Y = y$, $Z = z, \ldots$ occur.

Definition 4 (Probability density) *We call* probability density *of a stochastic variable X a function p_X that assigns to each proper interval $A := \{x_i : a \leq x_i \leq b\}$ with $a, b \in \mathbb{T}$, a value $p_X(A) \in [0, 1]$ representing the probability that X assumes one of the value in the interval A.*

Specifically, the computation of the probability density p_X varies depending on whether X is a categorical or a continuous variable. If X is a categorical variable, then p_X is computed as:

$$p_X(A) := \sum_{x_i \in A} P(X = x_i)$$

[6]Usually the measurable space $\mathbb{T}$ equivales the set of Real numbers. However, this is not always the case. For example, when we use stochastic variables to describe stochastic models, the adopted measurable space is generally the finite set of all possible states of the model.

Differently, if X is continuous variable, then p_X is computed as:

$$p_X(A) := \int_a^b P(X)$$

where a and b denote the extremes of the interval A.

1.5 Markov Models

After introducing the fundamentals of probabilities, we discuss now probabilistic models for describing the evolution over time of stochastic agents, i.e., agents that exhibit quantifiable uncertain behaviors. We focus in particular on Markov models, a specific typology of probabilistic models playing a central role in the domain of model checking and formal verification. Notable examples of Markov models include discrete and continuous-time Markov chains, Markov reward models, Markov decision processes, hidden Markov models *et cetera* [105]. For the purposes of the present work, we limit the discussion to discrete-time Markov chains (Section 1.5.1), Markov reward models (Section 1.5.3) and Markov decision processes (Section 1.5.4).

1.5.1 Markov Chains

Let $\mathcal{S}$ be a finite non-empty set of possible states. We are interested in modelling stochastic agents that, at each discrete-time $t \in \mathbb{N}$, shift from a state $s \in \mathcal{S}$ to another ,not necessarily different, state $s' \in \mathcal{S}$. We assume the stochastic behaviour of an agent to be time-homogeneous, that is, the probability of a transition from s to s' is independent of the time t at which it occurs, and memory-less, that is, the probability of each transition is independent of the previously occurred transitions. Under these conditions, the behaviour of the agent can be described in terms of a stochastic process called *discrete-time Markov chain* (DTMC) and defined as follows:

Definition 5 (Discrete-time Markov chain) *A discrete-time Markov chain M_{DTMC} is a tuple:*

$$M_{\mathsf{DTMC}} := \langle \mathcal{S}, T, \iota \rangle$$

where:

- *$\mathcal{S}$ is a finite non-empty set of states,*
- *$T : \mathcal{S} \times \mathcal{S} \to [0, 1]$ is a transition matrix that assigns a probability value $T(s, s')$ to each possible transition $s, s' \in \mathcal{S} \times \mathcal{S}$,*

- $\iota : \mathcal{S} \to [0,1]$ *is an initial probability distribution that assigns an initial probability value to each* $s \in \mathcal{S}$

Given a DTMC M_{DTMC}, we call *path* a function $\pi : \mathbb{N} \mapsto \mathcal{S}$ whose values are the states reached by M_{DTMC} at the various time-steps t. Accordingly, each path π describes a possible temporal evolution of the process and corresponds to an infinite countable sequence of states. In what follows, notation $\pi(t)$ is used to refer to the state of the path π at time t, while Paths(s) denotes the set of all paths π originating in a given state $s \in \mathcal{S}$ (i.e., such that $\pi(0) = s$). The set of all possible paths π of a given DTMC M_{DTMC} is denoted by $\Pi^{M_{\mathsf{DTMC}}}$ and represents the set of all possible outcomes of the process M_{DTMC}.

To relate paths with probabilities, we endow $\Pi^{M_{\mathsf{DTMC}}}$ with a σ-algebra and augment it to a probability space as follows. Given a path π of M_{DTMC}, let a *finite prefix* $\hat{\pi}$ of π be any sequence $(\pi(0), \ldots, \pi(t))$ originating in $\pi(0)$ and including a finite number of subsequent states of π. The set of all finite prefixes of a given path π is denoted by $pref(\pi)$, while the set of all finite prefixes $\hat{\pi}$ originating in a given state $s \in \mathcal{S}$ is denoted by $\text{Paths}_{fin}(s)$ (see, [10, sec. 10.1]).

Definition 6 (Cylinder set) *The* cylinder set $Cyl(\hat{\pi})$ *induced by a finite prefix* $\hat{\pi}$ *is defined as:*

$$Cyl(\hat{\pi}) := \{\pi \in \Pi^{M_{\mathsf{DTMC}}} \mid \hat{\pi} \in pref(\pi)\}\,.$$

That is, $Cyl(\hat{\pi})$ is the set of all paths whose common prefix is $\hat{\pi}$ [10, def. 10.9].

Definition 7 (σ-algebra of a Markov chain) *The σ-algebra associated with a DTMC M_{DTMC}, denoted $\sigma^{M_{\mathsf{DTMC}}}$, is the smallest σ-algebra that contains all cylinder sets $Cyl(\hat{\pi})$, where $\hat{\pi}$ ranges over all finite prefixes of M_{DTMC}.*

From basic concepts of probability (see, [10, sec. 10.1]) and definition 7, it follows that there exists a unique probability measure $P^{M_{\mathsf{DTMC}}}$ on the σ-algebra $\sigma^{M_{\mathsf{DTMC}}}$ such that:

$$P^{M_{\mathsf{DTMC}}}(Cyl(\hat{\pi}(0), \ldots, \hat{\pi}(t))) = \iota(\hat{\pi}(0)) \cdot P^{M_{\mathsf{DTMC}}}(\hat{\pi}(0), \ldots, \hat{\pi}(t))\,, \tag{1.1}$$

where:

$$P^{M_{\mathsf{DTMC}}}(\hat{\pi}(0), \ldots, \hat{\pi}(t)) := \prod_{\tau=0}^{t-1} T(\hat{\pi}(\tau), \hat{\pi}(\tau+1))\,, \tag{1.2}$$

while for finite prefixes composed by just one state (i.e., $\hat{\pi} = \{s\}$), $P_s(\hat{\pi}) = 1$ by definition [10, sec.10.1].

Also of interest are the probabilities of a path π, respectively a finite prefix $\hat{\pi}$, *conditional* on a given initial state $s \in \mathcal{S}$, which are henceforth denoted by $P_s^{M_{\mathsf{DTMC}}}(\pi)$, respectively $P_s^{M_{\mathsf{DTMC}}}(\hat{\pi})$.

Over the probability space $\langle \Pi^{M_{\mathsf{DTMC}}}, \sigma^{M_{\mathsf{DTMC}}}, P^{M_{\mathsf{DTMC}}} \rangle$, we define a family of categorical stochastic variables $\{S_t\}_{t \in \mathbb{N}}$ ranging over $\mathcal{S}$ that describe the temporal behaviour of the Markov chain. The above mentioned memory-less condition corresponds to the *Markov property*, stating that: $P^{M_{\mathsf{DTMC}}}(S_{t+1} \mid S_t, \ldots, S_0) = P(S_{t+1} \mid S_t)$. Time-homogeneity, on the other hand, corresponds to assume $P^{M_{\mathsf{DTMC}}}(S_{t+1} \mid S_t)$ to be the same for all t.

To conclude, let us consider the usual definition of (discrete-time) *stochastic process* .

Definition 8 (Stochastic process) *Given a finite non-empty set of states $\mathcal{S}$, a discrete-time* stochastic process *over $\mathcal{S}$, here denoted M, is a family of categorical stochastic variables $\{S_t\}_{t \in \mathbb{N}}$ ranging over $\mathcal{S}$ and defined over a probability space $\langle \Pi, \sigma(\pi), P^M \rangle$ such that:*

- Π *is the set of all paths generated by states in $\mathcal{S}$;*
- $\sigma(\Pi)$ *is the cylinder σ-algebra of Π;*
- P^M *is a probability measure over $\sigma(\Pi)$.*

From Definition 8, it follows that each stochastic process M is uniquely identified by (i.e., it is in one-to-one correspondence with) a probability measure P^M. Accordingly, a DTMC with state-space $\mathcal{S}$, initial distribution ι and transition matrix T can be alternatively regarded as the (discrete-time) stochastic process over $\mathcal{S}$ uniquely identified by the probability measure P^M generated by ι and T as by Equations (1.1) and (1.2).

Example 1 (Describing communication protocols with Markov chains) Consider a simple communication system operating with a single channel[7]. The channel is error-prone, meaning that messages can be lost. The are four possible states of the system: (1) *start*, (2) *try*, (3) *delivered*, and (4) *lost*. In state *start*, the system input the message and switches to state *try*. In state *try* the system tries to send the message. If the message is sent successfully, the system switches to the state *delivered* and then re-starts the cycle. Otherwise, it switches to state *lost* and then goes back to state *try*. At each further attempt, the probability to lose the message is 0.9.

We can model the behaviour of this system using a simple Markov chain $M_{\mathsf{DTMC}} := \langle \mathcal{S}, T \rangle$ such that: $\mathcal{S} := \{\text{start}, \text{try}, \text{lost}, \text{delivered}\}$ and T is specified as follows:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.1 & 0.9 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

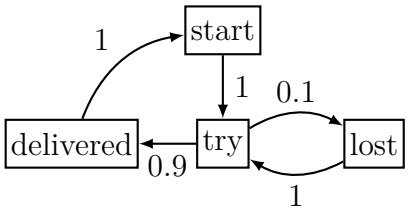


Figure 1.1: Transitions in a four-state DTMC.

Alternatively, we can visualize M using a directed graph whose edges are annotated by probabilities, as in the figure here below.

Through M we can ask interesting queries that arise with the stochastic behaviour of the system, such as: (i) What is the probability to lose the message at time-step 3?, (ii) what is the probability to lose the message whithin 10 time-steps?, or (iii) what is the probability to lose the message eventually in the future? The next section illustrates some strategies to answer these and similar queries computing relevant inferences in Markov chains.

1.5.2 Inferences in Markov Chains

In this paragraph, we recall two probabilistic inferences that are of central interest in this work, i.e., *marginal* and *hitting* probability.

Definition 9 (Marginal probability) *Given an event $B \subseteq \mathcal{S}$ and an initial state $s \in \mathcal{S}$, the marginal probability of B with respect to time t conditional on $S_0 = s$ is defined as follows:*

$$P_s^{M_{\mathsf{DTMC}}}(S_t \in B) := \sum_{\hat{\pi}:=\{\hat{\pi}(0),\ldots,\hat{\pi}(t)\} \text{ s.t. } \hat{\pi}\in \mathrm{Paths}_{fin}(s) \wedge \hat{\pi}(t)\in B} P^{M_{\mathsf{DTMC}}}(\hat{\pi})\,. \tag{1.3}$$

The next inference we recall is *hitting probability* (also called *reachability probability*, see [10]). Given an event B, the latter is informally defined as the probability of eventually reaching B starting from some initial state $s \in \mathcal{S}$, $\iota(s) > 0$. More rigorously, h_B can be defined as the probability that the distribution P^M assigns to the event "the process eventually reaches B" (in model checking literature, this is usually denoted by $\Diamond B$). The latter corresponds to a measurable set of paths given by the union of all cylinders $Cyl(\hat{\pi})$ spanned by finite prefixes $\hat{\pi}$ originating in s and such that $\exists\, t \in \mathbf{N} : \pi(t) \in B \wedge \forall \tau < t\, \pi(\tau) \notin B$. Since these sets are pairwise disjoint, we can rigorously define h_B as follows [10, sec.10.1.1.].

[7] This example is borrowed from [10, p.739]

Definition 10 (Hitting probability) *Given a DTMC M_{DTMC}, a set of states $B \subseteq \mathcal{S}$, and an initial state $s \in \mathcal{S}$,*

$$h_B(s) := \sum_{\hat{\pi} \in \mathrm{Paths}_{fin}(s):\, \exists\, t \in \mathbf{N} \text{ s.t. } \hat{\pi}(t) \in B \wedge \forall\, \tau < t,\, \hat{\pi}(\tau) \notin B} P_s^{M_{\mathsf{DTMC}}}(Cyl(\hat{\pi}))$$

Following [163] and [97], we define a procedure to compute marginal and hitting probabilities based on a *transition operator* $\hat{T}$ defined as follows.

Definition 11 (Transition operator) *For any real function f of $\mathcal{S}$, $\hat{T}f$ is defined as a function $\mathcal{S} \to \mathbf{R}$ such that:*

$$\forall s \in \mathcal{S}\,,\ (\hat{T}f)(s) := \sum_{s' \in \mathcal{S}} T(s, s') \cdot f(s')\,. \tag{1.4}$$

In practice, the transition operator returns the conditional expectation of f, i.e., $\hat{T}f(s) = E[f(S_{t+1}) \mid S_t = s]$ [97]. After t time steps, the transition operator is obtained as follows:

$$\forall s \in \mathcal{S}\,,\ (\hat{T}^t f)(s) := \begin{cases} (\hat{T}f)(s) & \text{if } t = 1 \\ (\hat{T}(\hat{T}^{t-1}f))(s) & \text{if } t > 1\,, \end{cases} \tag{1.5}$$

and the respective conditional expectation writes as $E[f(S_t) \mid S_0 = s] = (\hat{T}^t f)(s)$. Since the marginal probability is the expectation of the indicator function $\mathbb{I}_B(s)$ that returns one if $s \in B$ and zero otherwise, we can compute it as:

$$P_s(S_t \in B) = \hat{T}^t \mathbb{I}_B(s)\,. \tag{1.6}$$

The hitting probability h_B is obtained instead by computing the minimal non-negative solution[8] of the following system of linear equations [97, Corollary 19]:

$$h_B = \mathbb{I}_B + \mathbb{I}_{B^c} \cdot \hat{T} h_B\,, \tag{1.7}$$

where B^c is the complement of B, and sums and products are intended as element-wise operations on arrays. Standard methods solve Equation (1.7) in polynomial time with respect to $|\mathcal{S}|$ [10, p.749]. Here we consider an alternative procedure that is easier to be extended to the imprecise-probabilistic framework (see, Section 4.2). Let $h_B^t(s)$ be the probability of hitting B from $s \in \mathcal{S}$ *within a finite number of time-steps* t. For $t = 0$, we trivially have $h_B^t(s) = \mathbb{I}_B(s)$. For $t > 0$, if $s \notin B$, the hitting probability at t is obtained by applying the transition operator to h_B^{t-1}, while if $s \in B$ it is simply set to one. Thus:

$$h_B^t = \mathbb{I}_B + \mathbb{I}_{B^c} \cdot \hat{T} h_B^{t-1}\,. \tag{1.8}$$

[8] By minimal non-negative solution we mean a solution such that: (i) for each $s \in \mathcal{S}$, $h_B(s) \geq 0$, (ii) given any other non-negative solution $h'_B(s)$, then $h_B(s) \leq h'_B(s)$ for each $s \in \mathcal{S}$.

The hitting probability h_B can thus be computed as the fixed point of the iterative schema in Equation (1.8), as reported in [97, Lemma 14].

Notice that the time-complexity of computing h_B through the schema in Equation (1.8) results polynomial in $|\mathcal{S}| \cdot t^*$, where t^* denotes the number of iterations necessary to reach the fixed point. Indeed, each subsequent iteration of the schema requires to solve a linear equation system for each $s \in \mathcal{S}$ and, thus, its time-complexity results polynomial in $|\mathcal{S}|$. As t^* subsequent iterations are required to reach the fixed point, the overall time-complexity results polynomial in $|\mathcal{S}| \cdot t^*$.

1.5.3 Markov Reward Models

Among the various extensions of Markov chains, let us first consider *Markov reward models* (MRMs) [10]. A MRM is a pair $\langle M, rew \rangle$ composed of a Markov chain M with state space $\mathcal{S}$ and a reward function $rew : \mathcal{S} \mapsto \mathbb{N}$ such that $rew(s)$ represents the reward earned visiting s, for each state $s \in \mathcal{S}$. Given an event $B \subseteq \mathcal{S}$ and a path $\pi \in \Pi^{M_{\text{DTMC}}}$, we are interested in the cumulative reward earned along π until visiting an $s \in B$ for the first time. The latter is defined as follows.

Definition 12 (Cumulative reward) *Given an event $B \subseteq \mathcal{S}$ and a path $\pi \in \Pi^{M_{\text{DTMC}}}$,*

$$Rew_B(\pi) := \begin{cases} \sum_{\tau=0}^{t} rew(\pi(\tau)) & \text{if } \exists t : \pi(t) \in B \wedge \forall \tau < t,\, \pi(\tau) \notin B \\ \sum_{\tau=0}^{\infty} rew(\pi(\tau)) & \text{otherwise}. \end{cases} \tag{1.9}$$

Given the above definition, the *expected cumulative reward* earned until reaching B starting from $s \in \mathcal{S}$, denoted $ExpRew_B(s)$, is now defined as the expectation[9] of the function Rew_B conditional on initial state $s \in \mathcal{S}$, i.e., $ExpRew_B(s) := E[Rew_B \mid S_0 = s]$ [10, Def. 10.71]. Let us now discuss more in detail how to compute this value. First of all, we need to recall the following result from [10, Sec. 10.5.1]. Given an event $B \subseteq S$, let $\mathcal{S}^B_{=1}$ indicates the set of all states $s \in \mathcal{S}$ from which it is possible to reach an $s' \in B$ *almost surely*, i.e.:

$$\mathcal{S}^B_{=1} := \{ s \in \mathcal{S} \mid h_B(s) = 1 \}.$$

If $s \notin \mathcal{S}^B_{=1}$, then $ExpRew_B(s)$ may not converge to a finite value. Following [10, sec. 10.5.1], we thus assume for convenience that, by default, $ExpRew_B(s) = \infty$ for all $s \notin \mathcal{S}^B_{=1}$. For all $s \in \mathcal{S}^B_{=1}$, the following result holds [10, sec. 10.5.1].

[9] Notice that Rew_B is not a real-valued function, hence its expected value may be not defined for some arguments.

Proposition 2 *The values $x_s = E[Rew_B \mid S_0 = s]$ for each $s \in \mathcal{S}^B_{=1}$ provide the unique solution of the following equations system:*

$$x_s = \begin{cases} rew(s) & \text{if } s \in B, \\ rew(s) + \sum_{s' \in \mathcal{S}^B_{=1}} T(s,s') x_{s'} & \text{otherwise.} \end{cases} \tag{1.10}$$

There exist several methods to solve the linear system in Eq. (1.10) (see, [10, Sec.10.5.1]). Here, we adopt a recursive schema similar to that one involved above for hitting probability, which is obtained as follows.

For every $s \in \mathcal{S}^B_{=1}$, let $ExpRew^0_B(s) \coloneqq rew(s)$, and, for every $t \in \mathbb{N},\ t \neq 0$, let $ExpRew^t_B(s)$ be defined as:

$$ExpRew^t_B(s) \coloneqq \begin{cases} rew(s) & \text{if } s \in B\,, \\ rew(s) + \sum_{s' \in S} T(s,s') ExpRew^{t-1}_B(s') & \text{otherwise.} \end{cases} \tag{1.11}$$

Notice that the functions $ExpRew^t_B$ are well-defined since if $s \in \mathcal{S}^B_{=1}$, then $s' \in \mathcal{S}^B_{=1}$ for every s' such that $T(s,s') > 0$. Each function $ExpRew^t_B$ can be given a clear interpretation as the expected cumulative reward earned until reaching B from s *within* a maximum number of time-steps t, as the following result shows.

Theorem 3 *For every $t \in \mathbb{N}$, it holds that*

$$(\forall s \in \mathcal{S}^B_{=1})\ ExpRew^t_B(s) = E[Rew^t_B \mid S_0 = s], \tag{1.12}$$

where for every $\pi \in \Pi^{M_{\text{DTMC}}}$, $Rew^0_B(\pi) \coloneqq rew(\pi(0))$, and for every $t \in \mathbb{N},\ t \neq 0$,

$$Rew^t_B(\pi) \coloneqq \begin{cases} Rew_B(\pi) & \text{if } \exists t^* \leq t : (\forall \tau < t^*)\ \pi(\tau) \notin B, \pi(t^*) \in B, \\ \sum_{\tau=0}^{t} rew(\pi(\tau)) & \text{otherwise.} \end{cases} \tag{1.13}$$

Proof 1 *The statement is proven by induction. First, we prove the induction base:*

$$(\forall s \in S^B_{=1})\ E[Rew^0_B \mid S_0 = s] = rew(s) =: ExpRew^0_B(s). \tag{1.14}$$

Next, note that for every $t \in \mathbb{N},\ t \neq 0$, the value of Rew^t_B on any π is completely determined by the values $\pi(0), ..., \pi(t) \in S$. Thus, we can alternatively interpret Rew^t_B as a function on S^{t+1}. Moreover, for any $t \in \mathbb{N},\ t \neq 0$ and $s_0, ..., s_t \in S^{t+1}$, we observe that:

$$Rew^t_B([s_i]^t_{i=0}) = rew(s_0) + I_{B^c}(s_0) Rew^{t-1}_B([s_i]^t_{i=1})\,, \tag{1.15}$$

with $Rew^0_B(s_0) = rew(s_0)$. Exploiting this observation, we can proceed with the induction step. Assuming that the statement is true for $t-1$, with $t \in \mathbb{N},\ t \neq 0$, we prove that it is also true for t. First, for every $s \in S^B_{=1}$, we have that

$$\begin{aligned} &E[Rew^t_B([S_i]^t_{i=0}) \mid S_0 = s] = \\ &\quad = E[rew(s) + I_{B^c}(s) Rew^{t-1}_B([S_i]^t_{i=1}) \mid S_0 = s] = \\ &\quad = rew(s) + I_{B^c}(s) E[Rew^{t-1}_B([S_i]^t_{i=1}) \mid S_0 = s] = \\ &\quad = rew(s) + I_{B^c}(s) E[E[Rew^{t-1}_B([S_i]^t_{i=1}) \mid [S_i]^1_{i=0}] \mid S_0 = s], \end{aligned}$$

where in the last step we use the law of iterated expectations (or [97, Proposition 39] and [97, Proposition 4]). Clearly, $E[Rew_B^{t-1}([S_i]_{i=1}^t) \mid [S_i]_{i=0}^1]$ *does not depend on the initial state* S_0*, hence*

$$\begin{aligned} &E[Rew_B^{t-1}([S_i]_{i=1}^t) \mid [S_i]_{i=0}^1] = \\ &\quad = E[Rew_B^{t-1}([S_i]_{i=1}^t) \mid S_1] = \\ &\quad = E[Rew_B^{t-1}([S_i]_{i=0}^{t-1}) \mid S_0]. \end{aligned}$$

Now, $E[Rew_B^{t-1}([S_i]_{i=0}^{t-1}) \mid S_0 = s] = ExpRew_B^{t-1}(s)$ *for every* $s \in S_{=1}^B$ *by the inductive hypothesis. Hence, plugging this back into the expression we obtained for* $E[Rew_B^t([S_i]_{i=0}^t) \mid S_0 = s]$*, we have*

$$\begin{aligned} &E[Rew_B^t([S_i]_{i=0}^t) \mid S_0 = s] = \\ &\quad = rew(s) + I_{B^c}(s)E[ExpRew_B^{t-1}(S_1) \mid S_0 = s] = \\ &\quad = rew(s) + I_{B^c}(s)[\hat{T}ExpRew_B^{t-1}](s) =: ExpRew_B^t(s), \end{aligned}$$

for all $s \in S_{=1}^B$*, by using the fact that a transition matrix for an homogeneous Markov chain encodes* 1 *time step expectation [97, Section 2.1].*

Thanks to Theorem 3, we can now demonstrate the following main result proving that the recursive scheme above introduced converges to what we expect.

Theorem 4 $E[Rew_B \mid S_0]$ *restricted to* $S_{=1}^B$ *is a fixed point of the iterative scheme* (1.11)*.*

Proof 2 *First, notice that* $\lim_{t\to+\infty} Rew_B^t = Rew_B$*. Hence, for every* $s \in \mathcal{S}_{=1}^B$*:*

$$\lim_{t\to+\infty} E[Rew_B^t \mid S_0 = s] = E[Rew_B \mid S_0 = s]$$

by the monotone convergence theorem since $0 \leq Rew_B^0 \leq Rew_B^1 \leq ... \leq Rew_B$ *(for [97, Proposition 7] and [97, Proposition 4]). From Theorem 3, it follows that* $ExpRew_B^t$ *are also non decreasing and non negative. Hence, by using as well the continuity of* $\hat{T}$ *with respect to non decreasing and non negative sequences, we find that*

$$\begin{aligned} &E[Rew_B \mid S_0] \mid_{\mathcal{S}_{=1}^B} = \\ &\quad = \lim_{t\to+\infty} E[Rew_B^t \mid S_0] \mid_{\mathcal{S}_{=1}^B} = \lim_{t\to+\infty} ExpRew_B^t = \\ &\quad = \lim_{t\to+\infty} (rew_{\mathcal{S}_{=1}^B} + I_{B^c}\hat{T}ExpRew_B^{t-1}) = \\ &\quad = rew_{\mathcal{S}_{=1}^B} + I_{B^c}\hat{T}E[Rew_B \mid S_0] \mid_{\mathcal{S}_{=1}^B}, \end{aligned}$$

where the third and fifth step follow from Theorem 3, while the fourth step follows from the definition of $ExpRew_B^t$*.*

As for hitting probability, we can thus compute $ExpRew_B$ by iterating the schema in Equation (4.5) over increasing values of t until convergence.

The last MRM inference we consider is the *reward-bounded* hitting probability $h^r_B(s)$, i.e., the probability of reaching B from s before earning a cumulative reward equals to r.

As for standard hitting probability, we proceed by defining the event "the process reaches B before earning a cumulative reward equals to r" (usually denoted by $\Diamond_{\leq r} B$) as a measurable set of paths. Notably, this event corresponds to the union of all cylinder sets $Cyl(\hat{\pi})$ spanned by finite prefixed $\hat{\pi}$ originating in s and such that $\exists t \in \mathbb{N} : \hat{\pi}(t) \in B \wedge \forall \tau < t, \hat{\pi}(\tau) \notin B \wedge rew(\hat{\pi}(0), \ldots, \hat{\pi}(\tau)) \leq r$. Since these are pairwise disjoint sets, the reward-bounded hitting probability of B can be defined as follows [10, Sec. 10.5.1]:

Definition 13 (Reward-bounded hitting probability) *For all $s \in \mathcal{S}$:*

$$h^r_B(s) := \sum_{\substack{\hat{\pi} \in \mathrm{Paths}_{fin}(s) : \exists t \in \mathbb{N} \text{ s.t.} \\ \hat{\pi}(t) \in B \wedge \forall \tau < t\, \hat{\pi}(\tau) \notin B \wedge \\ rew(\hat{\pi}(0), \ldots, \hat{\pi}(t)) \leq r}} P^{M_{\mathsf{DTMC}}}_s(Cyl(\hat{\pi}))$$

In order to introduce a method to compute $h^r_B(s)$, let us first recall the following result from [10, Sec. 10.5.1]. Let h^ρ_B denote the vector of reward-bounded hitting probability for a reward threshold $\rho = 0, \ldots, r$. Let $\mathcal{S}^B_{>0}$ be the set of all states $s \in \mathcal{S}$ such that $h_B(s) > 0$, i.e., the set of all states $s \in \mathcal{S}$ such that there exists at least one path π originating in s and reaching B for some t.

Proposition 5 *For each $s \in \mathcal{S}$, the value of $h^\rho_B(s)$ is given by the following system of equations:*

$$h^\rho_B(s) = \begin{cases} 1 & \text{if } s \in B \text{ and } rew(s) \leq \rho \\ 0 & \text{if } rew(s) > \rho \text{ or } s \notin \mathcal{S}^B_{>0} \\ \sum_{s' \in \mathcal{S}} T(s, s') h^{\rho - rew(s)}_B(s') & \text{otherwise}. \end{cases} \tag{1.16}$$

Trivially, $h^\rho_B = 1$ whenever s is already in B and its state-reward does not overcome the desired threshold ρ. Differently, if $s \in B$ but $rew(s) > \rho$, then we have that the cumulative reward earned by the agent already overcomes the specified threshold and, thus, $h^\rho_B(s) = 0$. The same also holds when $s \notin \mathcal{S}^B_{>0}$ as, in this case, we already know that h_B is equal to zero and consequently also $h^\rho_B = 0$. In all the remaining cases, $h^\rho_B(s)$ is computed by iteration over the possible successors of s. That is, we take the sum over all $s' \in \mathcal{S}$ of the probability of reaching s' from s multiplied the probability of reaching B from s' before earning a cumulative reward equal to $\rho - rew(s)$. Remember that every time the agent transits from a state s to one of its successors $s' \in \mathcal{S}$, it earns the reward

of s. Consequently, the threshold ρ has to be reduced of a value $rew(s)$ at each further iteration from a state s to its successors s'.

Notice that the system in the above proposition is in fact a linear system with variables (s, ρ) ranging in $\mathcal{S} \times \{0, 1, \ldots, \rho\}$. As in the case of standard hitting (see, Eq. (1.16)), we can solve the system by standard methods [10, Sec. 10.5.1]. Here, we follow a different strategy based on a recursive schema that iterates both over *times* and *rewards*. Let $h_B^{t,\rho}(s)$ denote the probability of hitting B from s before earning cumulative reward ρ and within time-step t. For each $t \in \mathbb{N}$, the values of $h_B^{t,\rho}$ computed for $\rho = 0, \ldots, r$ are collected in $\mathcal{S} \times r$ matrix that we denote by $\mathbf{h}_B^{t,\rho_{0:r}}$.

For $t = 0$, we generate $\mathbf{h}_B^{t=0,\rho_{0:r}}$ by computing the vectors $h_B^{t=0,\rho}$ for $\rho = 0, \ldots, r$ and for each $s \in \mathcal{S}$ as follows:

$$h_B^{t=0,\rho}(s) = \begin{cases} 1 \text{ if } s \in B \text{ and } rew(s) \leq \rho \\ 0 \text{ otherwise}. \end{cases} \tag{1.17}$$

Consider that when $t = 0$ no transition occurs. Hence, we clearly have that, for each $s \in \mathcal{A}$, $h_B^{t=0,\rho}(s)$ is equals to *one* if s belongs to the hitting event B and its reward $rew(s)$ does not exceed the specified threshold ρ, otherwise $h_B^{t=0,\rho}(s)$ is *zero*.

For $t > 0$, $\mathbf{h}_B^{t,\rho_{0:r}}$ is generated by computing the various vectors $h_B^{t,\rho}$ for $\rho = 0, 1, \ldots, r$ via the following recursive schema applied to each $s \in \mathcal{S}$:

$$h_B^{t,\rho}(s) = \begin{cases} 1 & \text{if } s \in B \text{ and } rew(s) \leq \rho \\ 0 & \text{if } h_B(s) = 0 \text{ or } rew(s) > \rho \\ \sum_{s' \in \mathcal{S}} T(s, s') h_B^{t-1,\rho - rew(s)}(s') & \text{otherwise}. \end{cases} \tag{1.18}$$

The first two cases follow from considerations analogous to those leading to Eq. (1.17). In the third case, we obtain the value of $h_B^{t,\rho}(s)$ from $h_B^{t-1,\rho-rew(s)}$ via one-step application of the transition operator. As the reward is cumulative, the threshold ρ is decreased by the reward of the current state. The values of $h_B^{t-1,\rho-rew(s)}$ are provided by the matrix $\mathbf{h}_B^{t-1,\rho_{0:r}}$ that we generate at time-step $t-1$. Specifically, $h_B^{t-1,\rho-rew(s)}(s')$ corresponds to the cell identified by the $|s'|$-row and the $|\rho - rew(s)|$-column of the matrix $\mathbf{h}_B^{t-1,\rho_{0:r}}$. For all $s \in \mathcal{S}$, the procedure converges to a finite value, as proved by the following theorem.

Theorem 6 *Let $\langle M, rew \rangle$ be a MRM, $B \subseteq [\mathcal{S}]$, and $\rho \in \mathbb{N}$. There is a $t^* \in \mathbb{N}$ such that for all $\tau \geq 0$:*

$$\mathbf{h}_B^{t^*+\tau,\rho_{0:r}} = \mathbf{h}_B^{t^*,\rho_{0:r}}. \tag{1.19}$$

Proof 3 *It is actually enough to verify that, given a MRM (M, rew) and $B \subseteq [\mathcal{S}]$, for all $s \in \mathcal{S}$ and all $r \in \mathbb{N}$, it holds that:*

$$\exists t^* \in \mathbb{N} : \ \forall \tau \geq 0, \ h_B^{t^*+\tau,r}(s) = h_B^{t^*,r}(s). \tag{1.20}$$

Let $s \in \mathcal{S}$, $B \subseteq \mathcal{S}$ and $r \in \mathbb{N}$. We show that, for all $t \in \mathbb{N}$, the time-bounded hitting probability $h_B^{t,r}(s)$ is equivalent to the time-bounded hitting probability $h_{B^*}^t(s)$ of an alternative event B^* computed on the alternative MRM (M^*, rew^*), with $M^* := \langle \mathcal{S}^*, T^* \rangle$. Such structure is construct as follows. Let $r^* := r + \sup_{s' \in \mathcal{S}} rew(s')$. We first inductively define $\mathcal{S}^*$. Set $X_0 := \{(s, rew(s))\}$, and for every $n > 0$,

$$X_{n+1} := X_n \cup \{(s'', k) \in \mathcal{S} \times \mathbb{N} \mid \exists (s', m) \in X_n \; s.t. \; m \leq r \text{ and } k = rew(s') + m\}.$$

The idea is that each set $X_{n+1} \setminus X_n$ contains pairs (s', k) where s' is the last node belonging to a path of length $n-1$ starting in s and whose cumulated reward is $k \leq r^*$.
Notice that the sequence $(X_\ell : \ell \in \mathbb{N})$ is increasing monotone. Moreover for every n, $X_n \subseteq \{(s'', k) \in \mathcal{S} \times \mathbb{N} \mid k \leq r^*\}$, where the latter is a finite set. Hence, there is ℓ such that $X_\ell = X_{\ell+1}$, and define $\mathcal{S}^* := X_\ell \cup \{\bot\}$, where $\bot$ is some new "sink" state. For the remaining functions, we define, for $(s'', m), (s', k) \neq \bot$, $rew^*((s', k)) := rew(s')$ and

$$T^*((s', k), (s'', m)) := \begin{cases} T(s', s'') & \text{if } m = rew(s'') + k, \\ 0 & \text{else;} \end{cases}$$

whereas we set $rew^*(\bot) := 0$,

$$T^*(\bot, \circ) := \begin{cases} 1 & \text{if } \circ = \bot, \\ 0 & \text{else;} \end{cases}$$

and $T^*((s', k), \bot) := 1 - \sum_{(s'', m) \neq \bot} T^*((s', k), (s'', m))$. Thus, (M^*, rew^*) is essentially the tree unravelling around s of (M, rew) up to a cumulation of rewards bounded by r^*. That is, every finite path $\hat{\pi}$ in M starting in s such that $rew(\hat{\pi}) \leq r$ corresponds to a finite path $\hat{\pi^*}$ in M^* starting in $(s, rew(s))$ with $P_s(\hat{\pi}) = P_{(s,rew(s))}(\hat{\pi^*})$ and $rew(\hat{\pi}) = rew^*(\hat{\pi^*})$. Viceversa, every finite path $\hat{\pi^*}$ in M^* starting in $(s, rew(s))$ such that $rew^*(\hat{\pi^*}) \leq r$ corresponds to a finite path $\hat{\pi}$ in M starting in s with $P_s(\hat{\pi}) = P_{(s,rew(s))}(\hat{\pi^*})$ and $rew(\hat{\pi}) = rew^*(\hat{\pi^*})$. Given this observation, let $B^* := \{(s'', k) \in \mathcal{S}^* \mid s'' \in B, k \leq r\}$. We first notice that for every $t \in \mathbb{N}$:

$$h_B^{t,r}(s) = h_{B^*}^{t,r}((s, rew(s))) = h_{B^*}^t((s, rew(s))). \tag{1.21}$$

***Claim**: Let $s' \in \mathcal{S}$. Assume there is a finite path $\hat{\pi}(0) \dots \hat{\pi}(\ell)$ with $\hat{\pi}(0) = s$, $\hat{\pi}(\ell) = s'$, $k := rew(\hat{\pi}) \leq r^*$, then Equation (1.21) is actually a consequence of the following list of facts:*

1. *If $h_B(s') = 0$ or $rew(s') > r$ then $h_B^{t,\rho}(s') = h_{B^*}^{t,\rho}((s', k)) = h_{B^*}^t((s', k)) = 0$, for every $\rho \leq r$ and every $t \in \mathbb{N}$,*
2. *Else, if $k > r$, then $h_{B^*}^{t,\rho}((s', k)) = h_{B^*}^t((s', k)) = 0$, for every $\rho, t \in \mathbb{N}$,*

3. Else for every $t \in \mathbb{N}$

$$h_B^{t,(r+rew(s')-k)}(s') = h_{B^*}^{t,(r+rew(s')-k)}((s',k)) = h_{B^*}^{t}((s',k)). \tag{1.22}$$

Proof of Claim. *The first two items being clear (e.g. the second is simply due to the fact that $(s',k) \notin B^*$ and $h_{B^*}((s',k)) = 0$), we verify the last one. First notice that $k = rew(\hat{\pi}) \leq r$. This means that $s' \in B$ if and only if $(s',k) \in B^*$. We now reason by induction on t. For $t = 0$, we have that all terms are either equals to 1 or 0. However, given that $k \leq r$ and thence $rew(s') = rew^*((s',k)) \leq r + rew(s') - k$, the identities hold. We now verify the induction step, and therefore suppose that for $t \in \mathbb{N}$, $h_B^{t,r+rew(s'')-m}(s'') = h_{B^*}^{t,(r+rew(s'')-m)}((s'',m)) = h_{B^*}^{t}((s'',m))$, for every $(s'',m) \in \mathcal{S}^*$ such that $h_B(s'') \neq 0$, $rew(s'') > r$ and $m \leq r$. Let $\theta := r + rew(s') - k$. Hence $rew(s') \leq \theta$. Also, remember that by hypothesis $h_B(s') > 0$, meaning that*

$$h_B^{t+1,\theta}(s') = \begin{cases} 1 & \text{if } s' \in B \\ \sum_{s'' \in \mathcal{S}} T(s',s'') h_B^{t,\theta-rew(s')}(s'') & \text{otherwise}. \end{cases}$$

If $s' \in B$, reasoning as before, Equation (1.21) holds. We thus consider the case when $s' \notin B$, and terms $h_B^{t,\theta-rew(s')}(s'')$. Let $(s'',m) \in \mathcal{S}^*$, with $m - rew(s'') = k$, meaning that $h_B^{t,\theta-rew(s')}(s'') = h_B^{t,r-k}(s'') = h_B^{t,r+rew(s'')-m}(s'')$. Notice that by definition $T(s',s'') = T^*((s',k),(s'',m))$. Therefore, if we prove that $h_B^{t,r+rew(s'')-m}(s'') = h_{B^*}^{t,(r+rew(s'')-m)}((s'',m)) = h_{B^*}^{t}((s'',m))$ we are done. Now, let $\epsilon := r - k \geq 0$, as $k \leq r$. Therefore, $r + rew(s'') - m = r - k = \epsilon$. We reason by cases. If $h_B(s'') = 0$ or $rew(s'') > r \geq \epsilon$, then $h_B^{t,\epsilon}(s'') = h_{B^*}^{t,\epsilon}((s'',m)) = h_{B^*}^{t}((s'',m)) = 0$. Else, assume $m = rew(s'') + k > r$. This means $rew(s'') > r - k = \epsilon$, and therefore $h_B^{t,\epsilon} = (s'') = h_{B^*}^{t,\epsilon}((s'',m)) = h_{B^*}^{t}((s'',m)) = 0$. Else, $m = rew(s'') + k \leq r$, and conclude by induction hypothesis.
We finally end the main proof by applying Equation (1.21) to the fact that, by the convergence of standard hitting probability, there is $t^* \in \mathbb{N}$ such that, for every $\tau \geq 0$, $h_{B^*}^{t^*+\tau}((s,rew(s))) = h_{B^*}^{t^*}((s,rew(s)))$.

As for standard hitting, we can therefore compute h_B^r simply by iterating $\mathbf{h}_B^{t,\rho_{0:r}}$ over increasing values of t until convergence.

1.5.4 Markov Decision Processes

Markov decision processes (MDP) are an extension of Markov chains that allow for representing both probabilistic and non-deterministic behaviours [10]. The non-deterministic behaviours concern the actions that agents can choose, while the probabilistic behaviours concern the different possible outcomes that follow from the chosen action. To be clearer,

consider a simple experiment involving an agent who is asked to choose between rolling a die or tossing a coin[10]. This experiment has eight possible outcomes, the six faces of the die plus the two sides of the coin. The final outcome depends on both the probability distributions governing the stochastic behaviour of the die, respectively, the coin, and the non-deterministic choice of the agent. If the agent chooses to roll the die, then the probability of each face to be the final outcome is one sixth. Contrarily, if the agent chooses to toss the coin, then the probability of each side of the coin to be the final outcome is one half. This is an example of a kind of scenarios that can be modelled via MDPs.

Definition 14 (Markov decision process) *A MDP M_{MDP} is a tuple $\langle \mathcal{S}, \mathcal{A}ct, T, \iota, R \rangle$ where:*

- *$\mathcal{S}$ is a finite non-empty set of states;*
- *$\mathcal{A}ct$ is a finite non-empty set of actions;*
- *$T : \mathcal{S} \times \mathcal{A}ct \times \mathcal{S} \to [0, 1]$ is a transition matrix that assigns to each tuple $\langle s, a, s' \rangle$ the probability of the agent to transit from s to s' given that it performs action a;*
- *$\iota : \mathcal{S} \to [0, 1]$ is an initial probability distribution over states;*
- *$R : \mathcal{S} \times \mathcal{A}ct \times \mathcal{S} \to \mathbf{R}$ is a reward function that assigns to each tuple $\langle s, a, s' \rangle$ a real number representing the reward earned by the agent in selecting action a when in state s and thus transits to state s'.*

A MDP works as follows. At each discrete-time $t \in \mathbf{N}$, the agent chooses non-deterministically an action $a \in \mathcal{A}ct$ and then shifts from its actual state $s \in \mathcal{S}$ to one of the possible successors $s' \in \mathcal{S}$ of s that are allowed by a, i.e., such that $T(s, a, s') > 0$. Once selected an action and completed a transition, the agent gets the reward associated with the transition. The reward is cumulative, that is, the reward an agent gets after t time-steps iterations is the sum of all the rewards it earned from the initial state until the actual one.

Paths and Probability Measures

Differently from Markov chains and Markov reward models, a *path* π in an MDP M_{MDP} is defined as a function $\mathbb{N} \to \mathcal{S} \times \mathcal{A}ct$ that assigns to each time $t \in \mathbb{N}$ a pair $\langle s, a \rangle \in \mathcal{S} \times \mathcal{A}ct$ whose elements represent, respectively, the state of the agent at time t and the action selected by the agent in that state. Accordingly, each path is in one to one correspondence with an infinite countable sequence $s, a, s', a', \ldots$ of alternating states and actions. As for

[10] Assuming that both are fair.

Markov chains, we write $\pi(t)$ to denote the state of path π at time t, while we use $\alpha_\pi(t)$ to denote the action selected by the agent when in $\pi(t)$. Finally, notation Paths(s) is used to denote the set of all paths π originating in a given state and $\Pi^{M_{\mathsf{MDP}}}$ is used to denote the set of all paths of a given MDP M_{MDP}. Analogously to Markov chains, we endow the set $\Pi^{M_{\mathsf{MDP}}}$ with a σ-algebra corresponding to the smallest σ-algebra generated by all cylinder sets $Cyl(\hat{\pi})$ of M_{MDP}. The pair $\langle \Pi^{M_{\mathsf{MDP}}}, \sigma(\Pi^{M_{\mathsf{MDP}}}) \rangle$ is thus augemented to a probability space by including a probability measure probability measure P^{MDP} such that:

$$(\forall s \in \mathcal{S})\ P^{M_{\mathsf{MDP}}} := \iota(s) \cdot P^{M_{\mathsf{MDP}}}(Cyl(\hat{\pi}(0), \ldots, \hat{\pi}(t))\,, \tag{1.23}$$

where

$$P^{M_{\mathsf{MDP}}}(Cyl(\hat{\pi}(0), \ldots, \hat{\pi}(t)) := \prod_{\tau=0}^{t} T(\hat{\pi}(\tau), \alpha_{\hat{\pi}}(\tau), \hat{\pi}(\tau+1))\,, \tag{1.24}$$

and for finite prefixes including only one state, $P^{M_{\mathsf{MDP}}}(Cyl(\hat{\pi})) := 1$.

Policies

A notion of fundamental relevance for MDPs is that of *policy*, sometimes also referred to as *scheduler* or *strategy* [154]. A policy $\theta : \mathcal{S} \times \mathcal{A}ct \to [0, 1]$ is a probability distribution that assigns to each pair $\langle s, a \rangle$ the probability of an agent to select action a in state s. A policy is called *deterministic* when it assigns only zero and one values. A deterministic policy, therefore, can be also regarded as a function $\theta : \mathcal{S} \to \mathcal{A}ct$ that assigns to each state $s \in \mathcal{S}$ an action $a \in \mathcal{A}ct$ that the agent selects in s. In this work, we focus only on deterministic policies [11]. Unless otherwise specified, we will always use the generic term "policy" to refer to non-deterministic policies.

The application of a policy θ to a MDP M_{MDP} resolves all the non-deterministic choices about actions and induces a unique probability distribution T^θ over the transitions. This distribution is fully specified by a transition matrix $T^\theta : \mathcal{S} \times \mathcal{S} \to [0, 1]$ whose values $T^\theta(s, s')$ for each $\langle s, s' \rangle \in \mathcal{S} \times \mathcal{S}$ correspond to $T(s, \theta(s), s')$, with $\theta(s)$ denoting the action that θ deterministically assigns to s. In practice, the application of a policy θ reduces an MDP M_{MDP} to a Markov chain with transition matrix T^θ.

The *expected discounted cumulative reward* relative to a policy θ and an initial state $s \in \mathcal{S}$[12], denoted by $V^\theta(s)$, is the expectation, conditional on s and computed with respect to the distribution T^θ, of the discounted sum of the rewards that an agent gets by applying

[11] For the reader interested in non-deterministic policies, we refer to [155].

[12] In reinforcement learning literature, this is typically called *Value Function*[154].

θ, i.e.:

$$V^{\theta}(s) := E_{\theta}[\sum_{t=0}^{\infty} \gamma^t \, R(s_t, \theta(s_t), s_{t+1}) \mid S_0 = s] \,, \tag{1.25}$$

where $\gamma \in [0, 1]$ is a discounting factor that ensures the convergence of the sum. For MDPs with finite states and actions [155], the value of $V^{\theta}(s)$ can obtained by solving the following equation for all $s \in \mathcal{S}$:

$$V^{\theta}(s) = \sum_{s' \in \mathcal{S}} T^{\theta}(s, s')[R(s, \theta(s), s') + \gamma V^{\theta}(s')]. \tag{1.26}$$

In general, MDPs are associated with an optimization task that consists of learning the policy that maximizes the $V^{\theta}(s)$ for all $s \in \mathcal{S}$. Such a policy is usually called the *optimal policy* and denoted by θ^*. More specifically, θ is an optimal policy if and only if, for all $\theta' \in M_{\mathsf{MDP}}$ and $s \in \mathcal{S}$, $V^{\theta}(s) \geq V^{\theta'}(s)$.

For fully-specified MDPs with finite states and actions, the task of learning an optimal policy V^{θ} can be solved via dynamic programming based on two algorithms, called, respectively, *Policy Evaluation* and *Policy Iteration*. The Policy Evaluation algorithm is used to compute iteratively the value of $V^{\theta}(s)$ for all $s \in \mathcal{S}$ as follows:

Algorithm 1: Policy Evaluation

Data: MDP $= (\mathcal{S}, \mathcal{A}ct, T, \iota, R)$, discounting factor γ, policy θ
Result: State-value function V^{θ}

```
Initialize V^θ to 0 for all states
while V^θ not converged do
    foreach s ∈ S do
        V^θ(s) ← Σ_{s'∈S} T^θ(s, s')[R(s, θ(s), s') + γV^θ(s')];
    end
end
```

The policy iteration algorithm, on the other hand, computes the optimal policy θ by solving iterated optimization tasks.

Algorithm 2: Policy Iteration

Data: MDP $= (\mathcal{S}, \mathcal{A}ct, T, \iota, R)$, discounting factor γ
Result: Optimal policy θ

```
Initialize policy θ arbitrarily and set V^θ to 0 for all states;
while θ not converged do
    V^θ ← PolicyEvaluation((S, Act, T, ι, R), γ, θ);
    foreach s ∈ S do
        θ(s) ← arg max_{a∈Act} Σ_{s'∈S} T(s, a, s')[R(s, a, s') + γV^θ(s')];
    end
end
```

In practice, the algorithm works as follows. It starts form an arbitrary policy θ and computes V^θ through the policy evaluation algorithm reported above. Then, it computes an "improved" policy θ' by solving, for each $s \in \mathcal{S}$, the following optimization task: $\arg\max_{a \in \mathcal{A}ct} \sum_{s' \in \mathcal{S}} T(s, a, s')[R(s, a, s') + \gamma V^\theta(s')]$. The procedure is repeated until the policy *converges*, i.e., it does not change from one step to the next one.

Policies with Goal States

In many applications, we are interested in learning a policy θ that allows an agent to reach a given event $B \subseteq \mathcal{S}$ of interest. For example, consider a scenario like the one depicted in Figure 1.2. There is a robot that moves within a grid of $|8 \times 8|$ cells and whose goal is

−2	−4	−1	−1	0	−2	🥪	−3
−2	−4	−3	0	−6	0	−6	−2
−3	−2	−6	−4	−2	0	0	−2
−4	−5	−2	−9	−4	−9	−6	🍲
−2	−4	0	−2	−8	−1	−4	0
−6	−4	−9	−1	−5	−9	−2	−4
−6	−4	−9	−2	−4	−2	−6	−1
🤖	−4	−6	−2	−6	−9	−4	−1

Figure 1.2: Robot explorative task

to reach a cell where the food is located (in this case, $\langle 7, 1\rangle$ and $\langle 8, 3\rangle$). Actions consist of possible moves (e.g., "turn left", "turn right", "go straight" etc.) and negative rewards (here reported in the cells) represent the cost of taking an action and reaching a certain cell. We are interested in finding the sequence of actions (i.e., the policy) that allows the

robot to reach the food while minimizing the expected cumulative cost (i.e., maximizing the expected negative reward). This optimization task is similar to the one presented in the previous section but with an additional requirement: we want the agent to stop when it reaches a goal state (e.g., the cell with food). To manage with this additional requirement, we proceed as follows. Let M_{MDP} be an MDP with transition matrix T and let $B \in \mathcal{S}$ be the goal event. As a first step, we apply a transformation $\delta : T \to \mathbf{T}$ mapping the original transition matrix T to a transition matrix $\mathbf{T}$ where all $s \in B$ are made absorbing[13]. This trick ensures us that the agent remains in the goal state after reaching it. By replacing T with $\mathbf{T}$ in Equation (1.25), we obtain the *expected cumulative reward of eventually reaching an $s' \in B$ by following policy θ*, which we denote by V_B^θ. Our optimization task becomes then to find a policy θ that maximizes V_B^θ. This optimization task can be solved simply by replacing the matrix T with $\mathbf{T}$ in the Algorithms 1 and 2.

Notice that an alternative strategy to make goal states absorbing consists of acting on the reward function $R : \mathcal{S} \times \mathcal{A}ct \times \mathcal{S}$ and setting $R(s, a, s') = -\infty$ for all $s \in B$ and $s' \neq s$. In such a way, all transitions outgoing from every $s \in B$ are practically blocked ensuring us that the agent will remain in a goal state after reaching it. This second strategy is particularly useful whenever a full specification of the transition matrix T is not available, as in the case of model-free reinforcement learning that we will discuss in Chapter 5.

[13]Remember that a state $s \in \mathcal{S}$ is said "absrobing" if and only if $T(s, s') = 0$ for all $s' \neq s$ and $T(s, s') = 1$ for all $s' = s$.

Chapter 2

Probabilistic Model Checking: a Primer

Abstract

This chapter provides a general overview of the research field of model checking, with a specific focus on probabilistic model checking for both single-agent and multi-agent systems. The chapter begins with an introduction to the philosophical debate that accompanied the birth and development of the formal verification research program in the 70s and 80s. It then presents the most relevant languages and techniques adopted in contemporary model checking, including the logics LTL, CTL, and PCTL, along with their related semantics and model-checking algorithms. Special attention is given to probabilistic model checking for both single-agent and multi-agent systems. In particular, the chapter focuses on two recently-developed frameworks for model-checking stochastic multi-agent systems (i.e., PCTLK and COGWED), which serve as the foundation for further developments and applications presented in the remaining chapters of the thesis.

2.1 Formal Verification

Program correctness is the problem of determining whether computer programs behave as they are intended to do [52]. Since the early days of computer science this has been one of central problems of the discipline, as well as of its philosophical reflections. In the XIX century, at the time of Charles Babbage and Ada Lovelace, programming was mostly conceived as a sort of mechanical activity consisting of translating high-level formulae into commands executable by a physical machine. With the development of the first generation of modern electronic calculators in the 1950s, scholars realized that programming was a much more difficult endeavor than expected and how high the risk of dangerous failures and unexpected behaviors in computational machines was. Some events made also clear to the public and governments the potentially disastrous scale of the consequences of

software malfunctions. On October 5, 1960, for example, an undetected software error in the United States ballistic-missile early-warning system risked bringing the world into the midst of a nuclear catastrophe. The system wrongly indicated a massive launch of Soviet missiles against the United States; it was only a matter of luck that the error was discovered preventing the US government to launch missiles against the Soviet Union starting a nuclear war. Less dangerous but of big media impact was the 1962 software bug that led to the self-destruction of the Mariner 1 spacecraft. These and similar events caused an increasing skepticism towards software engineering and demolished the reputation of programming as a job [156]. In academia, many scholars got convinced that the continuing failures in software development were largely due to the lack of solid theoretical foundations and the poor methodological standards of software engineering. Indeed, "programming in those days was much more a craft than a science with a sound body of theoretical foundations" [156, p. 61]. In this climate of general uncertainty and frustration, a group of academic-oriented computer scientists started to dedicate their efforts to make programming a well-founded mathematical discipline. The group was strongly inspired by the view of programming as a mathematical activity, a view that is rooted on the deep connection between the notions of *proof* and *mechanical computation* whose origin dates back to the early twentieth-century debate on the foundations of mathematics[1]. The fathers of modern mathematical logic were fascinated by the idea of expressing deductive mathematical reasoning in terms of a few simple axioms and mechanically applicable rules of inference. To this aim, they introduced the notion of *formal system* that would had later played a central role in the mathematical foundations of modern computer science [131]. A formal system is usually defined in terms of a tuple of a formal language, a finite set of axioms (premises accepted as true) and a finite non-empty set of rules. Proofs of a statement can be obtained within a formal system by deriving it "mechanically" from the axioms through finite iterations of the rules.

A strict analogy holds between the notion of *proof* in a formal system and that of *program*: both consist of finite iterations of a fixed set of rules, the former leading from a set of premises (axioms) to a desired conclusion, the latter leading from a given *input* to a desired *output*. In more formal terms, this intuitive analogy is better described in terms of the well-known *Curry-Howard-De Bruijn correspondence* establishing the existence of an isomorphism between programs and proofs in a formal systems[2] (see, [28, ch.7]). In the sixties, this result represented the starting point of the of *formal verification* research

[1]For an in-depth overview of the debate on the foundations of mathematics and its connections with computer science see, [28, Part I].

[2]Notice that this isomorphism holds only under the interpretation of programs as *abstract machines*, i.e., as exact mathematical objects. Under the interpretation of programs as physical processes occurring in a machine, the notion of isomorphism can no longer be applied. As a mathematical notion, indeed, the latter can be applied only between two mathematical entities and not between a mathematical and a physical entity. In the latter case, the above correspondence can be considered as nothing more than an

program, whose central ideas will be briefly summarized in the next paragraphs.

2.1.1 The General Idea of Formal Verification

In order to understand the general idea beyond formal verification, let us introduce a bit more structured formulation of the program verification problem (from [28, def. 83]).

Definition 15 (Program verification problem) *Given any program P, is it possible to prove whether P when executed on the appropriate input $I \in \mathcal{I}$ returns the intended output $O \in \mathcal{O}$, or in other words whether P satisfies its intended function $f : \mathcal{I} \to \mathcal{O}$?*

The problem is addressed in formal verification as follows. First, one constructs a mathematical model M_{P} of the program P by means of a suitable formalism. Second, one provides a formal specification S_{P} of the intended behaviour (or part of it) of P in terms of a formula in a suitable formal language $\mathcal{L}$. Third, one verifies that M_{P} satisfies the specification S_{P}, i.e., that M_{P}'s behavior conform to its intended function expressed by S_{P}. The program P is then considered to be correct if and only if M_{P} always satisfies S_{P}, i.e., if there are no cases where the behaviour of M_{P} violates the requirement expressed by S_{P}. The kind of formalism used to model the program and specify its intended behaviour varies depending on the specific formal verification paradigm adopted.

One of the earliest logical formalisms specifically developed to reason about program correctness was surely *Hoare's logic* (also known as *Floyd-Hoare's logic*) [83], proposed by the British computer scientists and father of formal verification sir Charles Anthony Richard Hoare in 1969[3]. The basic element of Hoare's logic is the *Hoare triple*, which specifies how the execution of a given program-routine C modifies the state of a computation.

Definition 16 (Hoare triple) *An Hoare triple is defined as a tuple:*

$$Q\{C\}R$$

where:

- *Q is a set of* preconditions *that specifies the initial states (input) of the program;*
- *C is a* command *that specifies the execution of a given program-routine;*
- *R is a set of* postconditions *that specifies the intended final states (output) of the program;*

intuitive analogy.

[3]For a contextualized overview of Hoare's formalism, see [28, Ch. 7]

Preconditions and postconditions are formulated by means of *assertions*, expressions of the form $\mathcal{Q}[x := t]$ where $\mathcal{Q}$ is a predicate and $x := t$ denotes the substitution of a free-variable x in the formula with a value t. In addition to assertions, Hoare's system of logic includes a set of axioms describing arithmetic definitions and operations, and four inference rules to specify commands of a program[4]. A proof of formal correctness in Hoare's logic is constructed by assuming a given set of preconditions and showing that the post-conditions obtained by applications of the commands specifying the program are as intended. Program correctness is then reinterpreted with respect to a pair of pre- and postconditions as follows (see, [28, def. 84].

Definition 17 (Hoare correctness) *A program is correct with respect to a pre- and postconditions specification if and only if its execution on a machine whose initial state satisfies the preconditions will result in a final state that satisfies the postconditions.*

Notice that this notion of correctness is partial because it holds only under the condition that the program terminates. Indeed, a proof of total correctness requires also a proof of termination, which however cannot be formulated in Hoare's logic and must be derived separately. This limitation was overcome in the 1970s with the development of the *predicate transformer semantics* by the Dutch computer scientist Edsger Dijkstra [51, ch.3], which allows for the formulation of proofs that guarantee both termination and correctness with respect to the specification.

In the years following the publication of Hoare's seminal paper, a variety of different formalisms was introduced to reason about program correctness, all based on the idea of constructing formal derivations emulating the program and verifying that such derivations effectively lead from given inputs to intended outputs. Notable examples include the *operational semantics*, first developed as a semantic for the imperative programming language *Algol 68* [111] and then imported in the field of formal verification by Dana Scott [146], the *denotational semantics* developed by Scott and Strachey in the early seventies [147], and the *Axiomatic semantics*, which essentially combines and extends the approaches proposed by Floyd [69] and Hoare [83] (see, [20]).

Early formal verification approaches such as the ones mentioned above were strongly inspired by a philosophical view of programming as an exact science grounded on deductive reasoning and mathematical methods, a view that the philosopher and computer scientist Matti Tedre has recently called *strong formal verificationsm* (henceforth, the *strong view*) [156, ch. 4]. The latter is made explicit in the four theses on the fundamental nature of computer science formulated by Sir C.A.R. Hoare [84]:

1. Computers are mathematical machines;

[4]Notice that several new rules have been introduced in the system after Hoare's seminal paper to specify commands and constructs for increasingly complex programs.

2. Computer programs are mathematical expressions;

3. A programming language is a mathematical theory;

4. Programming is a mathematical activity.

During the last decades of the XX century, the strong view gained lots of popularity among academics and resulted in an attitude of extreme confidence and excessive optimism towards the formal verification program. Proponents of this position believed in the intellectual superiority of formal methods over empirical methods of testing based on simulations. This perception of the intellectual superiority of mathematical proofs led them to become quite excessively optimistic about the potentiality of the instruments they proposed. Indeed, many of them were strongly convinced that formal methods could guarantee the reliability of computational systems without any doubt. The feeling was that "formal methods has progressed to the point where the serious programmer should be expected to prove his programs in the same sense that a mathematician is expected to prove his theorems" [156, p. 66] For example, in his 1969 seminal paper, Hoare writes that "when all the parts of a computer system have been proven correct, their behavior can be predicted with confidence limited only by the reliability of the electronics" [83]. In a nutshell, if a program is formally proved to be correct, then nothing can go wrong; this was the motto.

Between the 1960s and 1970s, the strong view led to a deep intellectual rift between theoretical computer scientists, mostly based in university departments, and software engineering practitioners, mostly working in the industry. The latter, indeed, looked at the formal verification research program with little interest, considering its results and methods far away from the real practice and incapable of providing answers to their needs. On the practical side, indeed, the computer science agenda was going to look "more and more like engineering agenda then like mathematical agenda" [156, p. 68]. Since the early 1970s, claims against the strong view began to emerge among scholars even within the same community of supporters of formal verification. In 1970, Cristopher Strachey, one of the inventors of denotational semantics, argued that formal verification has little adherence with the real practice of programming and "can't demonstrate to the software engineering people on a sufficiently large scale that what it is doing is of interest or importance to them"[5]. Formal verification proofs appeared long and complex even for very simple programs, as well as obscure and harder to understand than the programs themselves. Furthermore, by the end of the 1970s, the majority of formal verification methods had been applied only to small-scale examples and several scalability issues emerged as they were confronted with the increasing complexity of real software systems.

[5]Stracey's claim is reported in [135, p. 9]. The quotation here reported is borrowed from [156, p. 65].

A heated philosophical debate finally emerged in the 80s between supporters and detractors of formal methods. In particular, a series of critical objections were raised against some of the fundamental epistemological and methodological assumptions of the formal verification research program, leading to the shipwreck of the strong view. The next two paragraphs summarize the arguably most influential among the arguments advanced in the 80s against formal verification, known by the names of their proponents, respectively, as *De Millo-Lipton-Perlis argument* and *Fetzer's argument*.

2.1.2 De Millo-Lipton-Perlis Argument

The first of the argument we analyse was formulated by anti-formalist computer scientists Richard De Millo, Richard Lipton and Alan Perlis [45]. Published on the *Communications of the ACM* at the end of the 70s, the argument was able to reach a very wide audience starting the philosophical debate on formal verification of the following years. In a nutshell, the argument criticizes the thesis that formal verification can aid establishing confidence in the correctness of programs. It is grounded on two main assumptions

1. The construction and acceptance of a proof in the mathematical community is fundamentally a *social process*;

2. Mathematical proofs and proofs of program correctness are two very distinct animals.

In support of (1), De Millo and collegues emphasize how proofs in mathematics are not "beautiful abstract objects with an independent existence" [45, p.273] but rather *messages* aimed at convincing the mathematical community of the truth of a given proposition. When got convinced to have found a potential proof for a given theorem, a mathematician writes a sketch of it and presents it to their colleagues. If the latter find the sketch interesting and convincing, then the mathematician writes a polished version of the proof and makes it accessible to a wider audience (e.g., through publication in Journals). A lot of mathematicians survey the proof and check it for errors, if the majority of them get convinced of its relevance and correctness, then the proof becomes an established result of mathematics. This does not mean that the proof is correct without any doubt; the history of mathematics is full of proofs that stood for centuries and were eventually discovered to be wrong. According the the authors, this even means that it is *likely* to be correct and the statement it proves is *likely* to be true. Furthermore, even after their acceptance in the mathematical community, proofs do not remain stable but are constantly modified, polished, improved, translated in new formalisms etcetera[6]. In this regard, proofs that we can found in mathematical practice are very far from the formal proofs of logicians. They

[6]In this regard, De Millo and collegues cite the example of Euler's formula reported by Imre Lakatos in his "Proofs and Refutations" [102].

are full of "intuitive" non-mechanical passages and make an extensive use of analogical and other non-deductive forms of reasoning. Good mathematical proofs must be able to convince mathematical audience of the truth of the statement they want to prove and to do so they must be simple and easy to survey. On the contrary, formal proofs are usually complex and tedious, requiring thousands of passages also for very trivial theorems. The practical ineffectiveness of formal proofs in mathematics, argue De Millo and colleagues, is evident since Russell and Whitehead's *Principia Mathematica*, which represented the crowing and at the same time the deathblow of the formalist approach in mathematics [45, p. 272]. The formal derivations included in the *Principia*, although not going beyond the theorems of elementary arithmetic, required the two English mathematicians an enormous intellectual effort and resulted in three gigantic volumes whose publication, however, aroused very little interest in the mathematical community, with the exception of a few specialists interested in the formal reconstruction of mathematical reasoning. If Russell and Whitehead's method were adopted widely in the mathematical community, De Millo and collegues note, very little progress could have been done in the history of mathematics and we "would still be counting on our fingers" [45, p.272]. A very similar critique can be advanced for proofs of formal correctness. These, the authors underline, do not look like messages aimed at convincing the audience of the truth of a statement, they are not constructed to be read and understood, they "cannot be internalized, transformed, generalized, used, connected to other disciplines, and eventually incorporated into a community consciousness" [45, p. 275]. In other words, mathematical proofs and proofs of program correctness are two very distinct animals. For this reason, the authors maintain, the social mechanism that establishes confidence in mathematical proofs does not apply for proofs of program correctness and, thus, the analogy with mathematical practice underlying the formal verification program is doomed to fail.

There is a possible objection to this argument that De Millo and colleagues consider. This is based on what they call the *scaling up* argument, which roughly goes as follows [45, p.277]:

- **Assumption 1:** verifications of small algorithms and model programs are comparable to mathematical proofs: being surveyable, understandable and shareable among specialists, they are subjected to the same social mechanism mathematical proofs are subjected to;

- **Assumption 2:** Complex software systems are made up of nothing more than simple algorithms and model programs;

- **Conclusion:** Hence, the admittely unreadable verification of a complex software system can be obtained as the sum of the (readable and shareable) verifications of small systems constituting it.

The authors, while agree with (1), strongly reject (2) arguing that no-continuty holds between small toy-example programs and complex software systems used in the real-world: "no programmer would agree that large production systems are composed of nothing more than small algorithms and programs. Patches, ad-hoc constructions, bandaids and tourniquets, [...] it has been estimated that more than half the code in any real production system consists of informal structures that are by definition inverifiable" [45, p. 277].

In the conclusion of their article, De Millo and collegues dismiss the idea that real-world complex software systems can be provably correct without any doubt: the best we can ask of software systems is to be *reliable*, i.e., to work sufficiently well.

2.1.3 Fetzer's Argument

Nine years after the publication of the De Millo and colleagues' article, a different argument against the strong view was proposed by philosopher of science James H. Fetzer [66]. Fetzer's analysis moves from distinguishing four different senses in which the term "program" is used in the computer science domain:

1. Programs as algorithms;
2. Programs as encoding of algorithms;
3. Programs as encoding of algorithms that can be compiled;
4. Programs as encoding of algorithms that can be compiled and executed by a machine;

In the former two senses, the term "programs" refers to abstract-mathematical objects that have no causal power and do not perform any physical work. In the latter two senses, the term refers to concrete-physical objects that have causal power on the world and do physical work (e.g., changes the voltage in a circuit). This distinction, Fetzer argues, has crucial methodological implications insofar as while "results in logic and mathematics fall within the domain of deductive methodology and require *demonstration*, lawful and causal claims fall within the domain of empirical inquiry and require *inductive warrants*" [66, p. 1050]. In this regard, one can identify two different senses of the term "verification". One sense characterizes pure mathematics and logic in which "theorems are subjected to verification by deriving them [through step-by-step iterated applications of rules] from no-premises at all (within systems of natural deduction) or from primitive axioms (within axiomatic formal systems)" [66, p. 1050]. The other sense characterizes experimental science in which a conclusion can be derived from given premises without absolute certainty but only with a certain degree of probability. Fetzer calls the former *absolute verifiability* while use the term *relative verifiability* for the latter. The differences between the two are enormous. Indeed, while statement "that are verified in the absolute sense cannot be false

[...] conclusions that are verified in the relative sense can still be false [...] and run the risk of observational and experimental verification" [66, p. 1051] According to Fetzer, the fallacy of the strong view lies in its claim to apply the notion of absolute verifiability to physical objects (i.e., programs in their meanings 3 and 4) which are by their very nature subject only to relative verifiability. Formal verification methods work perfectly as far as they are applied to study relevant properties of abstract programs (meanings 1 and 2) but cannot apply to properties of their implementations (meanings 3 and 4). In this regard, notice that Fetzer's argument does not attack formal verification *per-se* but rather the strong view and its claim that computer science reduces to mathematics.

Before concluding, there are some considerations regarding Fetzer's argument worth discussing. Indeed, one might be tempted to argue that, by limiting the applicability of formal verification methods to abstract programs only, Fetzer's argument renders this research program practically useless. As Fetzer himself claims, abstract programs are nothing more than *models* or *blueprints* with very few things in common with their implementations: an abstract program "could no more be similar to an [implemented] program than a mathematical model of a rainstorm could be similar to a storm [...] and cause things to get wet" [66]. Hence, what kind of benefit can we get from studying an abstract algorithm? after all, what is of interest for us are the behaviors of the implemented programs, not of their blueprints: we want to prevent a missile warning system from signaling the presence of a nuclear warhead when it does not exist; we want a spam filter not to trash important emails, et cetera. If Fetzer is right, formal methods are not useful for these tasks and so what is the point of continuing to study and develop them?

There is a fundamental mistake in this reasoning that relies on a confusion, which also Fetzer commits, between *models* (or *blueprints*) and *specifications*. The difference between these two notions has been recently clarified in [164] by appealing to the distinction between *description* and *prescription*: models *describe* how a computational system is constructed while specifications *prescribe* how to construct it in order to satisfy its intended function[7]. The difference emerges clearly in the case of mismatch: if a mismatch occurs between a model and its target-system, is the model that has to be refined; on the contrary, if a mismatch occurs between a specification and its target-system, is the system that has to be refined. Since specifications have normative jurisdiction over their target-systems, it becomes of fundamental relevance to understand and verify their properties.

For *executable programs* that governs our IT systems, the specification is provided by abstract programs. The latter have normative jurisdiction on the former in the sense that executable programs are constructed according to what specified by abstract programs, which in turn are constructed according to what specified by the intended function expressing the desired I/O behaviour of the system. To verify that an IT system behaves as

[7] On this distinction, see also [6, sec. 2.2].

intended, two different verification tasks are required:

1. to test that the executable program actually governing the behaviour of the system correctly implements the abstract program;

2. to verify that the abstract program effectively realizes the intended function.

Both levels are fundamental in the building process of an IT system and while for (1) Fetzer's argument holds (as executable and abstract programs have two very distinct natures), for (2) it does not hold as abstract programs and functions are entities of the very same nature, i.e., mathematical entities.

Verifying that an abstract program is conformed with the intended function is exactly what formal verification can and should aim to do. Of course, this is not *sufficient* to ensure that the real-world system based on that program behaves correctly (this is, in a nutshell, what Fetzer's argument proves) but is nevertheless *necessary*. Indeed, to assume the abstract level to be irrelevant for determining the behaviour of a program can erroneously lead one to think that failures and unintended behaviors depend univocally on malfunctions of the physical mechanism (i.e., the physical machine) executing the program. Literature, however, is full of examples of so-called *errors of design*, i.e., miscomputations that ultimately depends on the way the structure and functioning of the system are specified at the abstract level, see [71, 131].

To conclude, the lesson we can draw from Fetzer's analysis is that a proper study of computational systems require to distinguish among different *levels of abstraction* in the description of a computational system, each one requiring appropriate dedicate methodologies to be analysed and understood. We will return on this point more in detail in Chapter 3.

2.1.4 Formal Verification Today

Throughout the 1980s, the debate between supporters and detractors of formal methods was very heated, sometimes leading to the rise of extreme and highly polarized positions[8]. At first, the arguments of the anti-formalists severely undermined the credibility of the formal verification programme, which saw its popularity starting to decline and fundings being cut. The crisis was resolved starting from the end of the 80s with an in-depth rethinking of the methodologies and objectives of formal verification. The legacy of the debate is an increasing awareness that a complementary approach to the problem of program correctness is necessary: an approach getting together formal and experimental methods

[8]For a general overview of this debate, see [156, Ch. 4]. For a review of more contemporary arguments in favour and against formal methods, see [6].)

and considering both the abstract and implementation aspects that fundamentally characterize the nature of computational systems. Almost all scholars nowadays agree that the view of computer science as a purely mathematical discipline was reductive. Instead, computer science is a complex field with multiple methodological and epistemological "souls" (or "foundations", if you prefer): one close to mathematics, one close to engineering, and one close to experimental sciences[9]. The debate did not lead to the end of the formal verification program, which is nowadays more flourishing and expanding than ever, but it did imply an in-depth rethinking of both its conceptual and its methodological basis. The strong view has been abandoned in favor of weaker positions, roughly centered on the idea of formal verification as a means to increase the reliability of computational systems by discovering errors of design and by checking their safety and fairness. A variety of new techniques have been proposed to close the "dual" abstract and physical nature of computational systems. A notable example are *Separation logics* and their relative semantics, which allow for reasoning about how computational systems allocate and use their finite resources (notably, in terms of memory) [122, 90, 139]. Another example is represented by the development of *Model Checking* techniques based on temporal modal logics and the related Kripke-like possible world semantics [167]. This represents the main object of analysis of the present work of thesis and deserve an in-depth introduction, which will be carry out in the remaining sections of this chapter.

2.2 On Model Checking

Since the 1990s, the field of formal verification has been receiving a renewed boost thanks to the rise of *Model Checking*, a research program whose aim consists of developing fully-automatized procedures to check the correctness and reliability of computational systems. Typically, a model-checking procedure requires two inputs:

- a model of the target-system, generally in the form of a state-transition system whose states represent possible configurations of the target-system and transitions model its evolution across time;
- a specification of the desirable properties of the target-system in some formal language, usually some kind of computationally-grounded temporal logic.

Model checking procedures then operate by performing an exhaustive exploration of the system's state space and checking for each state whether it satisfies the specified desirable properties. In such a way, it is possible to identify anomalies, bugs and faults of the system

[9] An in-depth philosophical analysis of the three foundations of computer science and their mutual epistemological, methodological, and historical relationships has been recently proposed in [131].

before its implementation in real-life scenarios. Compared to more traditional formal verification techniques, model checking has proved to be more effective in facilitating the early detection of defects and reducing the verification time [10, p.7] for many different classes of computational systems. For this reason, it nowadays represents one of the most popular and expanding sectors of applied logic having successful applications in different fields, including: software verification [16], communication protocols [15], and even computational biology [22, 14].

The following sections provides an introduction to model checking, starting from its basic frameworks and then focusing on recent probabilistic and multi-agent developments. Section 2.3 provides an overview of transition systems, the mathematical structures typically used in model checking to specify the behaviour of computational systems. Section 2.5 introduces the fundamentals of model checking based on *Linear Temporal Logic* (LTL) and the related transition-systems semantics. Section 2.6 discusses the most known evolution of LTL, namely the *Computation Tree Logic* (CTL), and the related model-checking algorithms. Section 2.7 is dedicated to *probabilistic model checking*, the specific sub-branch of model checking focusing on the analysis of systems exhibiting stochastic behaviours. Notably, it introduces *Probabilistic Computation Tree Logic* (PCTL), the reference formalism to model-check stochastic systems against probabilistic properties whose semantics and related model-checking procedures are based on the Markov models formalism introduced in Chapter 1. To conclude, Section 2.8 outlines recent developments and applications of model checking to multi-agent systems, focusing in particular on epistemic extensions of CTL based on so-called *Interpreted Systems*. The first extension considered is CTLK [113], a logic originally conceived to specify both temporal and epistemic properties of multi-agent systems. The second one is PCTLK [175], a probabilistic-epistemic extension of CTL semantically based on *Probabilistic Interpreted Systems* [167]. The third extension we consider is the COGWED logic [28], a formalism also based on probabilistic interpreted systems and whose language is obtained by extending CTLK with a *weighted-doxastic* operator to specify agents' *degrees of beliefs*. These two multi-agent formalisms will represent the starting point for the subsequent developments and applications of probabilistic model checking and Markov models semantics presented in the rest of the dissertation.

2.3 Transition Systems

A fundamental requirement for model checking is a formalism to model the behavior of computational systems under analysis. The reference formalism used for this purpose is typically that of *transition systems* (TS). In very general terms, we can define a TS as a directed graph whose nodes represent possible *states* of a target-system and edges model possible transitions, i.e., state changes [10, p.19]. Each state describes the information

relative to the system at a specific time of its evolution. For example, in a TS modelling a traffic light the possible states are: *red*, *yellow*, and *green*. Transitions, on the other hand, specify possible evolutions of the system across time. Intuitively, if the TS admits only one possible temporal evolution, then we call it *deterministic*. Otherwise, we call it *non-deterministic*.

Definition 18 (Transition system (TS)) *A TS is a tuple* $\langle \mathcal{S}, Act, \rightarrow, I, AP, L \rangle$ *where:*

- $\mathcal{S}$ *is a finite non-empty set of states (called* state space*)*
- Act *is a set of actions*
- $\rightarrow \subseteq \mathcal{S} \times Act \times \mathcal{S}$ *is a transition relation*
- $I \subseteq S$ *is a set of initial states*
- AP *is a set of atomic propositions in a given language* $\mathcal{L}$
- $L : \mathcal{S} \rightarrow 2^{AP}$ *is a labelling function*

Notice that, for convenience, $s \rightarrow^a s'$ is generally used as a short notation for the transition $(s, a, s') \in \rightarrow$. Given two states s, s' and an action a, if $s \rightarrow^a s'$, then s' is called an *a-successor* of a. The tuple includes all the elements necessary to specify the behaviour of the TS across time. Let t be a variable ranging over $\mathbb{N}$ and whose values denote specific time-steps in the evolution of the system. At $t = 0$, the TS is in a given initial state $s \in I$. It then evolves across time by performing specific actions $a \in Act$ that allow it to pass from a state s to its *a-successor* s' according to what specified by the transition relation $\rightarrow$. If more than one possible transition exists for some $s \in \mathcal{S}$, then the TS chooses the transition *non-deterministically*. The evolution of the TS continues until a *terminal state* is reached, where the latter is defined as a state $s \in \mathcal{S}$ such that, for each $s' \in \mathcal{S} : s \neq s'$, no transitions $s \rightarrow s'$ are allowed.

The labelling function l assigns to each state $s \in \mathcal{S}$ a set of atomic propositions $l(s) \subseteq AP$ representing the *elementary properties* that the TS satisfies in that state. Thereby, it induces a satisfiability relation $\models$ for atomic formulae $p \in AP$ such that, given a TS and a state $s \in \mathcal{S}$:

$$TS, s \models p \text{ iff } p \in l(s) \tag{2.1}$$

The introduction of this satisfiability relations lays the foundations for using TS as semantic models for propositional logic languages suitable to specify desirable properties of TSs. Let us further clarify this point with the following simple example.

Example 2 (Beverage vending machine) We present an example of a simple TS to model the behaviour of a beverage vending machine. The example is borrowed from [10, p.21].

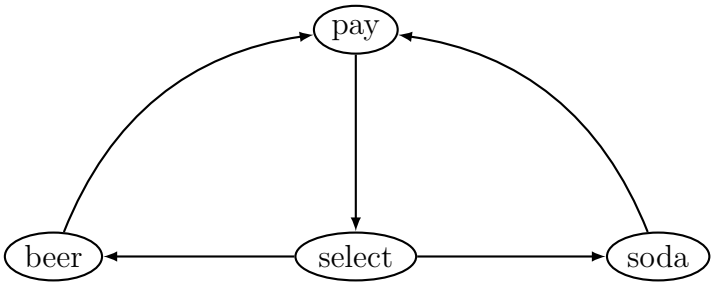


Figure 2.1: TS model of a beverage vending machine.

The TS has state space $\mathcal{S} := \{pay, select, soda, beer\}$, there is only one possible initial state, i.e., $I := \{pay\}$, and the set of actions is:

$$Act := \{insert\_coin, choose\_soda, choose\_beer, get\_soda, get\_beer\}$$

The TS's behaviour is deterministic in all states $s \in \mathcal{S}$ except that for *select*, in which two transitions are possible (i.e., $select \rightarrow soda$ and $select \rightarrow beer$) depending on which action the user selects among *choose_soda* and *choose_beer*.

The choice of the atomic propositions AP generally depends on the desirable properties one is interested to verify. For example, let suppose to be interested in checking whether the TS satisfies the following property when initialized in state *pay*:

$$P := ``\text{the vending machine delivers a drink only after providing a coin}''$$

In this case, a two-elements set of atomic propositions $AP := \{paid, drink\}$ is sufficient to describe the scenario at stake. The labelling function l we will use, therefore, will take the following form:

$$l(pay) = \emptyset,\ l(soda) = l(beer) = \{\text{paid}, \text{drink}\},\ l(select) = \{\text{paid}\}$$

To check that TS satisfies the above property, we have to verify that $s \models paid$ holds for all the states $s \in \mathcal{S}$ reached by TS starting from the initial state *pay* before reaching an $s' \in$ such that $s' \models drink$. At this point, the problem is to obtain a feasible procedure that explores the TS and verifies that it satisfies the mentioned condition. Before to obtain such a procedure, we need to define a way to specify complex desirable properties, such as the one mentioned in this example, in a compact form. As we will explain in the following sections, this is usually obtained by means of specific computationally-grounded formal languages derived from modal (propositional) temporal logic.

2.3.1 Predecessors and Successors

Consider a $TS := \langle \mathcal{S}, Act, \rightarrow, I, AP, l \rangle$, a state $s \in \mathcal{S}$ and an action $a \in Act$. We say that a state $s' \in \mathcal{S}$ is a *direct a-successor* of s if and only if TS can reach s' from s in one time-step by performing action a. Consequently, the set of all the direct a-successors of s, denoted by $Post(s, a)$, will be given by:

$$Post(s, a) := \{s' \in \mathcal{S} \mid s \rightarrow^a s'\}$$

Hence, the set of all the direct successors, i.e, the successors with respect to all the possible actions $a \in Act$, of state $s \in \mathcal{S}$, denoted by $Post(s)$, will be given by:

$$Post(s) := \bigcup_{a \in Act} Post(s, a)$$

Similarly, if $s' \rightarrow^a s$ holds, then we say that s' is a *direct a-predecessor* of s. The set of all the a-predecessors of s, denoted by $Pre(s, a)$ is then given by:

$$Pre(s, a) := \{s' \in \mathcal{S} \mid s' \rightarrow^a s\}$$

and, thus, the set of all the direct predecessors of s, denoted by $Pre(s)$, will be given by:

$$Pre(s) := \bigcup_{a \in Act} Pre(s, a)$$

Notice that if $Pre(s) = \emptyset$, then s is an *initial state*. On the other hand, if $Post(s) = \emptyset$, then s is a *terminal state*.

2.3.2 Executions, Paths and Traces

In the previous paragraph, we mentioned that a TS can have different evolutions over time. However, we did not provide this term with a precise mathematical meaning. Technically speaking, we can represent the possible evolutions of a TS across time in terms of *executions*.

Definition 19 (Execution fragment) *Given a $TS := \langle \mathcal{S}, Act, I, \rightarrow, AP, l \rangle$, we call* execution fragment *of TS, denoted by $\hat{\delta}$, a either finite or infinite alternating sequence of states and actions:*

$$\hat{\delta} := s, a, s', a', s'', \ldots$$

An execution fragment is called *maximal* if and only if either it is infinite or it is a finite execution fragment that ends in a terminal state. An execution fragment, instead, is called *initial* if an only if it originates from an initial state.

Definition 20 (Execution) *An* execution*, denoted by δ, is an initial and maximal execution fragment.*

Intuitively, executions are full specifications of the possible evolutions across time of TS that provide details on both the states visited by the system and the actions performed. In some cases, such full specifications are difficult to achieve as the actions may strictly depend on the users' intentions and not on the system's internal structure (see the example above 2). For this reason, it is common in model-checking to abstract away from the details about actions and consider TS without specifications for them:

Definition 21 (Transition systems (without actions)) *A TS (without actions), denoted by M_{TS} is a tuple $TS := \langle \mathcal{S}, \rightarrow, I, AP, l \rangle$, where all elements are defined as in Definition 18.*

We use notation M_{TS} to denote TSs without actions and distinguish them from standard TSs above denoted by TS. In the case of TSs without actions, the notion of execution is replaced by those of *path* and *trace*.

Definition 22 (Path fragment (TS)) *Given a $M_{TS} : \langle \mathcal{S}, I, \rightarrow, AP, l \rangle$, we call* path fragment *of M_{TS} a either finite or infinite sequence of states*

$$s_0, s_1, s_2, \ldots$$

induced by the temporal evolution of M_{TS} and such that $s_i \in Post(s_{i-1})$ for each $i > 0$.

A path-fragment is *maximal* if and only if it is either infinite or ends in a terminal state, while it is *initial* if it starts in an initial state $s \in I$.

Definition 23 (Path (TS)) *A* path *is a both maximal and initial path fragment*

The notion of *path* introduced in Definition 23 is the analogous for TSs of the homonymous notion introduced for Markov chains in Chapter 1. For this reason, we denote it using the same notation π.

Similarly, we use notation $\hat{\pi}$ to denote a *finite prefix* of a path π, here defined as a finite path-fragment $s_0, \ldots, s_t$ such that $s_0 \in I$ an s_t is a non-terminal state.

In some cases, the states $s \in \mathcal{S}$ of a TS are not directly observable but only are their atomic properties AP. In such cases, instead of considering paths $\pi := \pi(0), \pi(1), \ldots$, we consider their *traces*, i.e., the *words* composed by the finite or infinite concatenation of their labels.

Definition 24 (Word) *Given a $M_{TS} := \langle \mathcal{S}, I, \rightarrow, AP, l \rangle$, a word σ over the finite alphabet 2^{AP} is an infinite concatenation of symbols $A_0, A_1, \ldots$ such that $A_\tau \in 2^{AP}$ for each $\tau \in \mathbb{N}$. Analogously, a finite word $\hat{\sigma}$ is a finite concatenation of symbols $A_0, A_1, \ldots, A_t$ such that $A_\tau \in 2^{AP}$ for each $\tau \leq t : t \in \mathbb{N}$.*

Definition 25 (Trace) *Given a $M_{TS} := \langle \mathcal{S}, I, \rightarrow, AP, l \rangle$ and a path $\pi := \pi(0), \pi(1), \ldots,$ the* trace *induced by π, denoted by $trace(\pi)$, is the (finite or infinite) word $\sigma := A_0, A_1, \ldots$ defined over 2^{AP} such that $A_0 = l(\pi(0)), A_1 = l(\pi(1)), \ldots$*

With a slight abuse of terminology, given a state $s \in I$, we call *trace* of s each trace generated by a path π that originates in s. We denote by $Trace(s)$ the set of all the possible traces of s, i.e., the traces generated by all the paths π originating in s. Similarly, we denoted by $Trace(M_{TS})$ the set of traces of all the possible initial states $s \in I$, i.e.:

$$Traces(M_{TS}) := \bigcup_{s \in I} Traces(s)$$

2.3.3 Reachability

Two fundamental notions in model checking that we will make extensive use of in this work are those of *reachability* and *almost-sure reachability* of an event $B \subseteq \mathcal{S}$.

Intuitively, we say that an event B is *reachable* by M_{TS} if and only if M_{TS} can reach at least one $s' \in B$ eventually in the future starting from some initial state $s \in I$. With a slightly abuse of notation, we write $M_{TS} \models \Diamond B$ to denote that an event B is reachable by a given TS M_{TS}. In formal terms, reachability is defined as follows:

Definition 26 (Reachability (paths)) *Given a transition system M_{TS} and an event $B \subseteq \mathcal{S}$, the condition $M_{TS} \models \Diamond B$ holds if and only if there exists at least one path π in M_{TS} such that $\pi(t) \in B$ for some $t \in \mathbb{N}$.*

That is, an event B is reachable by a TS M_{TS} if and only if there exists at least one possible temporal evolution of M_{TS} leading to an $s' \in B$. An equivalent formulation of the notion of reachability can be obtained also by referring to traces instead of paths:

Definition 27 (Reachability (traces)) *Given a TS M_{TS} and an event $B \subseteq \mathcal{S}$, the condition $M_{TS} \models \Diamond B$ holds if and only if there exists at least one trace $\sigma := A_0, \ldots, A_t, \cdots \in Traces(TS)$ such that $A_t = l(s')$ for some $t \in \mathbb{N}$ and $s' \in B$.*

Alongside the notion of reachability, we find the cognate notion of *almost-sure reachability*, which we denote by $M_{TS} \models \Box B$ and define as follows:

Definition 28 (Almost-sure reachability) *Given a TS M_{TS} and an event $B \subseteq \mathcal{S}$, the condition $M_{TS} \models \Box B$ (B is almost-surely reachable by M_{TS}) holds if and only if, for all paths π in M_{TS}, there exists a $t \in \mathbb{N}$ such that $\pi(t) \in B$.*

That is, an event B is almost-surely reachable by a TS M_{TS} if and only if all the temporal evolutions (paths) of M_{TS} eventually lead to a state $s' \in B$. Similarly to reachability, an alternative definition for almost-sure reachability can be provided in terms of traces.

2.4 Linear Time Properties

A fundamental class of desirable properties that we are usually interested to analyse is that of *linear-time properties* (LTP), which we can roughly define as properties that specify what traces a TS should exhibit [10, p. 97]. This section provides a basic definition of LTPs as requirements on traces of a TS, while the reference formal language to specify LTPs, i.e., the so-called *Linear Temporal Logic* (LTL), will be introduced in the next section.

Consider a finite set of atomic propositions AP and its power-set 2^{AP}, we denote by $(2^{AP})^{\omega}$ the set of all *words* definable over 2^{AP}. With this notation, we can introduce a first definition of LTP as follows:

Definition 29 (Linear time property) *A* Linear Time Property *over 2^{AP} is a proper subset of $(2^{AP})^{\omega}$*

That is, a LTP is a *language* (i.e., a structured collection) of infinite words defined in the alphabet 2^{AP} [10, p. 98]

Definition 30 (Satisfaction relation for LTP) *Let P be an LTP defined over 2^{AP}. Given a TS M_{TS} and a state $s \in \mathcal{S}$, the following conditions hold:*

$$M_{TS} \models P \text{ iff } Traces(M_{TS}) \subseteq P$$
$$M_{TS}, s \models P \text{ iff } Traces(s) \subseteq P$$

That is, a TS M_{TS} satisfies a LTP P if and only if all the traces that it can generate are included in P. In other words, LTPs impose constrains on the desirable linear behaviours of a transition system.

Example 3 Let us go back to the example of the beverage vending machine reported above (2). Consider again the desirable property:

$$P := ``\text{the vending machine delivers a drink only after providing a coin}''$$

We can express this property as an LTP P that constrains the possible behaviours of our TS. In particular, P will be the set of all words of the form:

$$\emptyset\, paid\, paid \ldots\, drink \ldots$$

that is, all the words beginning with $\emptyset$, including a finite repetition of the proposition "*paid*" and then an instance of the proposition "*drink*". To check whether our TS satisfies P, hence, we have first to determine $Traces(M_{TS})$ and thus check whether $Traces(M_{TS}) \subseteq P$.

2.4.1 Safety and Liveness Properties

Among the various kinds of LTPs, particularly relevant to verify the reliability of computational systems are *safety* and *liveness* properties. In very general terms, a safety property P_{safe} is a LTP stating that "a bad event never happens", while a liveness property P_{live} is a LTP stating that "a good event will surely happen".

Consider the system reported in Example 3. An instance of safety property that can be formulated for this system is: "the machine never delivers a drink before the user has paid", where the *bad event* is "the machine delivers a drink before the user has paid". Differently, an instance of liveness property is: "the machine always delivers a drink after the user has paid", where the *good event* is "the machine delivers a drink after the user has paid".

Let B denote a "bad event", we can define the notion of *bad prefix* as follows.

Definition 31 (Bad prefix) *We call* bad prefix *relative to a bad event $B \subseteq \mathcal{S}$ a finite word $\hat{\sigma} := A_0, A_1, \ldots, A_t$ such that $A_\tau = l(s')$ for some $s' \in B$ and $\tau \leq t : t \in \mathbb{N}$*

In practice, a bad prefix is the finite trace generated by a path reaching the "bad" event B within a finite time-horizon $t \in \mathbb{N}$.

A safety property with respect to a given bad event B is the maximal set of words (i.e., the maximal LTP) that do not include any bad prefix relative to B. Formally, we can define a safety property as follows [10, def. 3.22].

Definition 32 (Safety property) *A* safety property P_{safe} *defined over a finite alphabet* 2^{AP} *is a LTP such that: for all infinite words* $\sigma \in (2^{AP})^{\omega} \setminus P_{safe}$ *there exists a finite prefix* $\hat{\sigma}$ *of* σ *such that*

$$P_{safe} \cap \{\sigma' \in (2^{AP})^{\omega} \mid \hat{\sigma} \text{ is a finite prefix of } \sigma'\} \neq \emptyset\,.$$

A liveness properties typically requires that a good event will happen infinitely often. Differently from safety properties, thus, liveness properties do not constrain the finite behaviours of a TS but impose specific conditions on the infinite behaviours. This intuition is captured by the following definition [10, def. 3.33].

Definition 33 (Liveness property) *A LTP* P_{live} *defined over a finite alphabet* 2^{AP} *is a* liveness property *if and only if, for all finite words* $\hat{\sigma} \in (2^{AP})^{*}$*, there exist an infinite word* $\sigma := A_0, A_1, \ldots$ *such that* $\hat{\sigma}$ *is a finite prefix of* σ *and* $\sigma \in P_{live}$*.*

2.5 Linear Temporal Logic

The reference language to specify LTPs is the *Linear Temporal Logic* (LTL), originally proposed in [129]. In this section, we first introduce the syntax and semantics of LTL and then present specific procedures to model-check transition systems against LTL formulae based on so-called *Non-deterministic Büchi Automata* (NBA).

Definition 34 (LTL Syntax)

$$\phi := \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \bigcirc\phi \mid \phi_1 \bigcup \phi_2$$

The language includes the standard notation $\top$ for propositional tautologies, the meta-variable p denoting atomic propositions, standard Boolean connectives for negation and conjunction, and two temporal modalities whose informal reading is the following:

- $\bigcirc\phi$: "ϕ holds in the next time-step"
- $\phi_1 \bigcup \phi_2$: "ϕ_1 holds in the next time-steps until ϕ_2 holds"

The introduction of the two new modal operators $\bigcirc$, informally called *next*, and $\bigcup$, informally called *until*, is motivated by the necessity of modelling LTPs in a compact form.

2.5.1 LTL Semantics

In general, the first step to introduce a semantics for LTL based on TSs consists of defining specific satisfiability conditions for LTL formulae relative to words:

Definition 35 (Satisfiability of LTL formulae (over words)) *Given a word* $\sigma := A_0, A_1, \ldots$ *defined over* 2^{AP}*. The following conditions hold:*

$$\sigma \models \top \ \forall \sigma \in (2^{AP})^{\omega}$$
$$\sigma \models p \text{ iff } p \in A_0$$
$$\sigma \models \phi_1 \wedge \phi_2 \text{ iff } \sigma \models \phi_1 \text{ and } \sigma \models \phi_2$$
$$\sigma \models \neg\phi \text{ iff } \sigma \not\models \phi$$
$$\sigma \models \bigcirc\phi \text{ iff } \sigma[1, 2 \ldots] \models \phi$$
$$\sigma \models \phi_1 \bigcup \phi_2 \text{ iff } \exists j \geq 0 \text{ s.t. } \sigma[j, \ldots] \models \phi_2 \text{ and } \forall\, 0 \leq i < j \ \sigma[i, \ldots] \models \phi_1$$

where $\sigma[1, \ldots]$ *is the suffix of* σ *starting in* A_1*, i.e.,* $\sigma[1, \ldots] = A_1, A_2, \ldots,$ *and* $\sigma[j, \ldots]$ *and* $\sigma[i, \ldots]$ *are the suffixes of* σ *starting in* A_j*, respectively,* A_i*.*

The subsequent step consists of introducing the set $Words(\phi)$, defined as the set of all words in $(2^{AP})^{\omega}$ that satisfy ϕ:

$$Words(\phi) := \{\sigma \in (2^{AP})^{\omega} \mid \sigma \models \phi\}$$

Notice that the set $Words(\phi)$ is a LTP. Specifically, it is possible to prove that $Words(\phi)$ is the one and only LTP induced by the given LTL formula ϕ (see, [10, p. 232-233]). Therefore, we can apply to $Words(\phi)$ the satisfiability condition for LTPs as in Definition 30:

$$M_{TS} \models Words(\phi) \text{ iff } Traces(M_{TS}) \subseteq Words(\phi)$$
$$M_{TS}, s \models Words(\phi) \text{ iff } Traces(s) \subseteq Words(\phi)$$

from which we can finally derive the specific satisfiability conditions for LTL formulae, as follows:

Definition 36 (Satisfiability of LTL formulae (over TS and states))

$$M_{TS} \models \phi \text{ iff } Traces(M_{TS}) \subseteq Words(\phi)$$
$$M_{TS}, s \models \phi \text{ iff } Traces(s) \subseteq Words(\phi)$$

2.5.2 LTL Model Checking

The model-checking problem for LTL consists of defining an automatic procedure that, given a transition system M_{TS} and a LTL formula ϕ, checks whether $M_{TS} \models \phi$. As a preliminary remark, notice that transition systems modelling real-life computational systems are usually very big and their size makes "manual" executions of model-checking tasks practically unfeasible. Automatic procedures are then fundamental. This section briefly presents the automatic procedures specific for LTL formulae. The latter are based on a particular class of abstract state machines called *Non-deterministic Büchi Automata* (NBA).

Definition 37 (NBA) *A NBA, denoted $\mathcal{A}ut$, is a tuple $\langle \mathcal{Q}, \Sigma, \delta, Q_0, \mathcal{F} \rangle$ where:*

- *$\mathcal{Q}$ is a finite non-empty set of states*
- *Σ is an alphabet*
- *$Q_0 \subseteq \mathcal{Q}$ is the set of initial states*
- *$\delta : \mathcal{Q} \times \Sigma \to 2^{\mathcal{Q}}$ is a transition relation*
- *$F \subseteq \mathcal{Q}$ is a set of* accept*, or final, states called the* acceptance set

Given a word $\sigma := A_0, A_1, \ldots$ defined in the alphabet Σ, we call *run* of σ, denoted by $run(\sigma)$, the path $\pi = q_0, q_1, \ldots$ such that $trace(\pi) = \sigma$. A run is *accepting* if and only if $q_i \in F$ for infinitely many indices $i \in \mathbb{N}$. The *accepted language* of $\mathcal{A}ut$, denoted by $\mathcal{L}(\mathcal{A}ut)$ is the set of all the words σ defined in the alphabet Σ whose run is accepting.

NBAs and LTL formulae are connected via the following theorem [10, Theorem 5.41]:

Theorem 7 (NBA for LTL formulae) *For any LTL formula ϕ defined over 2^{AP}, there exists a NBA $\mathcal{A}ut_\phi$ such that $Words(\phi) = \mathcal{L}(\mathcal{A}ut_\phi)$ and $\mathcal{A}ut_\phi$ can be constructed in time and space $2^{\mathcal{O}(|\phi|)}$, where $\mid \phi \mid$ denotes the lenght of the formula ϕ.*

This theorem guarantees that to build a NBA whose accepted language is $Words(\phi)$ is always possible. A detailed proof of the theorem and an explanation of the procedure to generate a NBA for a given LTL formula is available in [10, p.280-281].

Consider now the following observations:

$$
\begin{aligned}
&M_{TS} \models \phi \text{ iff } Trace(M_{TS}) \subseteq Words(\phi) \\
&\text{iff } Trace(M_{TS}) \cap Words(\neg(\phi)) = \emptyset \\
&\text{iff } Trace(M_{TS}) \cap \mathcal{L}(\mathcal{A}ut_{\neg(\phi)}) = \emptyset
\end{aligned}
$$

where $\mathcal{A}ut_{\neg(\phi)}$ is a NBA that accepts the LTP $\neg(\phi)$, meaning that there exists a word $\sigma \models \neg(\phi)$ whose run $run(\sigma)$ in $\mathcal{A}ut_{\neg(\phi)}$ is accepting. These observations prove that the model-checking task for an LTL formula ϕ effectively reduces to the task of checking whether the intersection set $Trace(M_{TS}) \cap \mathcal{L}_{\neg(\phi)}(\mathcal{A}ut)_{\neg(\phi)}$ is empty. In this regard, [10, p.198] proves the following propositions[10]:

Proposition 8 (Product automaton law) *$Trace(M_{TS}) \cap \mathcal{L}_{\neg(\phi)}(\mathcal{A}ut_{\neg(\phi)}) = \emptyset$ if and only if $\exists\, run(\sigma) \in M_{TS} \otimes \mathcal{A}ut_{\neg(\phi)}$ such that $\sigma \models \neg\phi$*

where, $M_{TS} \otimes \mathcal{A}ut_{\neg(\phi)}$ is the *product automaton* of the transition system M_{TS} and the NBA $\mathcal{A}ut_{\neg(\phi)}$ as by Definition 38.

Definition 38 (Product automaton) *Let M_{TS} be a TS without terminal state and $\mathcal{A}ut := \langle \mathcal{Q}, \delta, Q_0, F, 2^{AP} \rangle$ be a NBA. The product automaton of M_{TS} and $\mathcal{A}ut$ is defined as the tuple $M_{TS} \otimes \mathcal{A}ut := \langle \mathcal{S} \times \mathcal{Q}, Act, \rightarrow', I', AP', l' \rangle$ where:*

- *$\mathcal{S} \times \mathcal{Q}$ is a set of states obtained as the Cartesian product of the state spaces of, respectively, TS and $\mathcal{A}ut$*
- *$\rightarrow'$ is a transition relation defined by the rule $\langle s, q \rangle \rightarrow' \langle s', q' \rangle$ iff $s \rightarrow^{a} s' \wedge q \rightarrow^{l(s)} q'$*
- *$I' := \{\langle s_0, q_0 \rangle \mid s_0 \in I \wedge \exists q_0 \in Q_0 \,\text{s.t.}\, q_0 \rightarrow^{l(s_0)} q\}$*
- *$AP' = Q$*
- *$l' : \mathcal{S} \times \mathcal{Q} \rightarrow 2^{Q}$ is a labelling function such that $l'(\langle s, q \rangle) = \{q\}$*

In conclusion, the observations 2.2, jointly with propositions 7 and 8, prove that the model-checking task for a given LTL formula ϕ effectively reduces to the task of checking whether there exists a run $run(\sigma)$ in the product automaton $TS \otimes \mathcal{A}ut_{\neg(\phi)}$ such that $\sigma \models \neg(\phi)$. Interestingly, the latter is a task that can be executed in time polynomial in $\mid \mathcal{S} \times \mathcal{Q} \mid$ [10] and hence easily implementable into an automatic procedure. This is the key property to define the main algorithm for checking whether a TS satisfies a given LTL formula ϕ, which is reported in Figure 3.

The algorithm works as follows. After taking in input a TS M_{TS} and an LTL formula ϕ, it proceeds by building a NBA $\mathcal{A}ut_{\neg\phi}$ accepting formula $\neg\phi$, and deriving the product automaton $M_{TS} \times \mathcal{A}ut_{\neg\phi}$. Hence, it checks whether there exists a run in the product

[10]Notice that, in the proof provided by [10, p.198], the condition "there exists a run $run(\sigma)$ in $TS \otimes \mathcal{A}ut_{\neg(\phi)}\sigma \models$ such that $\sigma \models \neg\phi$" is expressed in terms of the satisfiability of a *safety property*. Here, we omit reference to safety properties as an in-depth discussion of them will arguably require too much space and goes beyond the purposes of the present work. The interested reader can refer to [10, ch. 3-4].

Algorithm 3: LTL Model Checking

Input: M_{TS} and ϕ
Output: "yes" if $M_{TS} \models \phi$; "no" plus a counter-example if $M_{TS} \not\models \phi$

```
1 Construct an NBA Aut_¬φ such that L(Aut_¬φ) = Words(¬φ)
2 Construct the product automaton M_TS ⊗ A
3 if ∃ run(σ) ∈ M_TS ⊗ A such that σ ⊨ ¬φ then
4 |   return "no" and a finite prefix of run(σ) showing that σ ⊨ ¬φ
5 end
6 if ∄ run(σ) ∈ TS ⊗ A such that σ ⊨ ¬φ then
7 |   return "yes"
8 end
```

automaton that accepts $\neg\phi$. If this is the case, then the algorithms returns "no" and provides a finite fragment of the run as a counterexample. Otherwise, the algorithm returns "yes". It is possible to prove that the overall complexity of the algorithm is $\mathcal{O}(\mid \mathcal{S} \mid \cdot 2^{|\phi|})$, where $\mid \mathcal{S} \mid$ denotes the cardinality of the transition system's state space and $\mid \phi \mid$ denotes the lenght of the LTL formula ϕ (see [10, p. 282]).

2.6 Computation Tree Logic

The LTL has been conceived specifically to model linear time properties and implicitly assumes a universal quantification over paths, that is, it assumes that a LTL formula ϕ holds in a state $s \in \mathcal{S}$ if and only if it holds *for all* the paths originating in s [10, p.309]. This assumption follows from the LTL semantics introduced in Section 2.5.1 and turns to be very problematic if we generalize to desirable properties that are not LTPs. For instance, consider the property: "for every executions, it is possible that the system returns to an initial state" [10, p.309]. This property cannot be specified in LTL language[11].

To overcome these issues, a new logic was introduced in the 80s, the so-called *Computation Tree Logic* (CTL) [30]. Differently from LTL, CTL is based on a *branching* notion of time, i.e., on the assumption that, for each state $s \in \mathcal{S}$ and time $t \in \mathbb{N}$, there are many possible successors state $s' \in \mathcal{S}$ that a transition system can reach at time $t+1$. This assumption entails that computations do not generate paths but *trees*, hence the name *computation tree* logic. In this section, we will present the CTL syntax (see, 39), its

[11] One might be tempted to specify this condition as $M_{TS}, start \models \Box\Diamond(\{start\})$, with *start* denoting the initial state and $\{start\}$ being an atomic proposition uniquely denoting the initial state. However, this requirement is too strong as it corresponds to say that "for every execution, the system *always* returns to the initial state", which is clearly different from what our desirable property states.

semantics (Section 2.6.1), and the related model-checking procedures (Section 2.6.2).

Definition 39 (CTL Syntax)

$$\phi := \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \exists\psi \mid \forall\psi$$
$$\psi := \bigcirc\phi \mid \phi_1 \bigcup \phi_2$$

The CTL language includes two distinct kind of formulae: *state formulae* ϕ, which represent properties of states, and *path formulae* ψ, which represent properties of *paths*[12]. ϕ formulae include the standard notation $\top$ for tautologies, Boolean negation and conjunction, and two operators to express *existential* and *universal* quantification over paths, whose informal reading is:

- $\exists\psi$: "there exists an outgoing path that satisfies ψ"
- $\forall\psi$: "All outgoing paths satisfy ψ"

On the other hand, ψ formulae include the already known modal operators for *next* and *until*. In this case, however, the meaning of the formula is related to path so that:

- $\bigcirc\phi$: "in the next state of the paths ϕ holds";
- $\phi_1 \bigcup \phi_2$: "ϕ_1 holds along the path until ϕ_2 holds".

Notice that ψ formulae are bounded by quantifiers so to allow the CTL language to express quantification over paths. For example, a formula like $\forall \bigcirc (p)$ holding in a given state $s \in \mathcal{S}$ expresses the complex property that "for all the paths π outgoing from s, the successor of s satisfies the atomic property p". Furthermore, indefinitely long nested formulae can be built by iteratively nesting ϕ and ψ formulae. In such a way, the CTL language can express a wide range of highly-complex properties that characterize contemporary computational systems.

2.6.1 CTL Semantic

The CTL semantics is based on the notion of *computation tree*. Given a TS M_{TS}, the computation tree of M_{TS} is a *directed acyclic graph*[13] whose vertices correspond to states $s \in \mathcal{S}$ and arrows correspond to possible transitions $s \to s' \in \mathcal{S} \times \mathcal{S}$. The computation

[12] Where each path π represents one of the possible realizations of a given computation of the TS.

[13] A directed graph is a pair $\langle V, \to \rangle$, where V is a set of *vertices* and $\to$ is a set of *arrows* connecting vertices. A directed *acyclic* graph is a graph that does not include cycles, i.e., paths that start and end in the same vertex $v \in V$.

tree represents all the possible evolutions across time of the M_{TS}[14]. Each single *path* in the tree (i.e., sequence of states $s_0, s_1, \ldots$ originating in a given initial state) represents a possible evolution of the system. Quantification over paths is then possible by looking at the paths outgoing from a given initial state and checking whether all ($\forall$), or at least one ($\exists$) path, satisfy a certain path property ψ. For simplicity, in what follows we use notation M_{TS} directly to denote the computation tree generated by a given transition system instead of the transition system properly.

For what concerns satisfiability conditions: in the case of ϕ formulae these are expressed with regard to (the computation tree of) a transition system M_{TS} and a state $s \in \mathcal{S}$, while, in the case of ψ they are expressed with regard to M_{TS} and a path π.

Definition 40 (Satisfiability of CTL formulae) *Given a transition system M_{TS} and a state $s \in \mathcal{S}$, respectively, a path π, the following conditions hold:*

$$
\begin{aligned}
&M_{TS}, s \models \top, \quad \forall s \in \mathcal{S}\,,\\
&M_{TS}, s \models p \ \text{ iff } \ p \in l(s)\,,\\
&M_{TS}, s \models \neg\phi \text{ iff } M_{TS}, s \not\models \phi\,,\\
&M_{TS}, s \models \phi_1 \wedge \phi_2 \ \text{ iff } \ M_{TS}, s \models \phi_1 \text{ and } M_{TS}, s \models \phi_2\,,\\
&M_{TS}, \pi \models \bigcirc\phi \text{ iff } M_{TS}, \pi(1) \models \phi\,,\\
&M_{TS}, \pi \models \phi_1 \bigcup \phi_2 \text{ iff } \exists \tau \geq 0 :\ \pi(\tau) \models \phi_2 \text{ and } \forall \tau' : 0 \leq \tau' < \tau, \pi(\tau') \models \phi_1\,.
\end{aligned}
$$

2.6.2 CTL Model Checking

Given a *finite*[15] transition system M_{TS}, an initial state $s \in \mathcal{S}$ and a CTL formula ϕ, the model-checking problem consists of defining a procedure to check whether $M_{TS}, s \models \phi$. The model-checking for CTL does not typically involve *traces* but work directly on set of states. Specifically, it is based on a set of procedures that iteratively enumerate the set $Sat(\phi) := \{s \in \mathcal{S} \mid M_{TS}, s \models \phi\}$, i.e., the set of all states $s' \in \mathcal{S}$ that satify a given formula ϕ, and hence checks whether $s \in Sat(\phi)$.

The standard model-checking algorithm for CTL and its extension is called *parsing tree*. This algorithm does not involve *traces* but work directly on set of states. More specifically, it exploits the *parse-tree* of Λ, which is the tree generated by decomposing Λ in its various sub-formula (see, Figure 2.2) [10, p. 336]. The algorithm works as follows:

1. It generates the *parse tree* of ϕ, recursively decomposing ϕ in its sub-formulae λ until only atoms remain;

[14]Notice that, if the TS admits more than one possible state, the computation tree is, in fact, a forest.
[15]The reason why we restrict our focus on just finite TSs are explained below.

2. It traverses the parse tree of ϕ visiting all the sub-formulae λ, starting from the leaves and working backwards to the roots,

3. At each sub-formula λ, it calculates $Sat(\lambda)$ (i.e., the set of states that satisfy λ) by checking whether $s \models \lambda$ for all $s \in \mathcal{S}$,

4. It calculates $Sat(\phi)$ by composition of the various $Sat(\lambda)$,

5. It finally checks whether $s \in Sat(\phi)$.

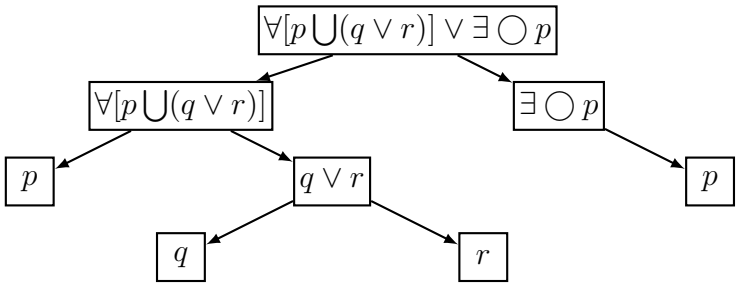


Figure 2.2: Parse-tree of a formula

The algorithm includes a specific sub-routine to compute $Sat(\lambda)$ for each specific kind of sub-formula λ of ϕ. For $\lambda := \top \mid \phi_1 \wedge \phi_2 \mid \neg\phi$, the algorithm computes $Sat(\lambda)$ by an iterative application of the recursive schema in Equation (4.4.1):

$$\begin{aligned} &Sat(\top) := \mathcal{S} \\ &Sat(p) := \{s \in \mathcal{S} : p \in l(s)\} \\ &Sat(\phi_1 \wedge \phi_2) := Sat(\phi_1) \cap Sat(\phi_2) \\ &Sat(\neg\phi) := \mathcal{S} \setminus Sat(\phi) \end{aligned} \tag{2.2}$$

For $\lambda := \exists\psi$, the procedure varies depending on the nature of the nested formula ψ:

- When $\psi := \bigcirc\phi$, the algorithm computes $Sat(\lambda)$ simply as $\{s \in \mathcal{S} \mid Post(s) \cap Sat(\phi) \neq \emptyset\}$.That is, $Sat(\bigcirc\phi)$ is determined as the set of all states that have at least one direct successors[16] satisfying ϕ.

- When $\psi := \phi_1 \bigcup \phi_2$, the algorithm follows the subroutine described in Figure 4. It generates the computation tree of M_{TS} and computes $Sat^t(\exists\phi_1 \bigcup \phi_2)$[17] for increasing

[16]Remeber that a direct successor of s in a given TS is a state $s' \in \mathcal{S}$ such that $s \rightarrow s'$ holds.

[17]This is defined as the set of all states from which the TS can reach an $s' \in Sat(\phi_1)$ in at most t steps via a ϕ_1-path fragment, that is, a path fragment whose states all satisfy ϕ_1.

Algorithm 4: $Sat(\exists \phi_1 \bigcup \phi_2)$

Input: A finite TS, a CTL formula $\exists \phi_1 \bigcup \phi_2$, $Sat(\phi)$ and $Sat(\phi_2)$
Output: $Sat(\exists \phi_1 \bigcup \phi_2)$

```
1  Sat^{t=0}(∃φ1 ⋃ φ2) ← {Sat(φ2)};
2  foreach t ≥ 0 do
3  |  if ∀s ∈ S : Post(s) ∩ Sat^t(∃φ1 ⋃ φ2) = ∅ then
4  |  |  Sat^t(∃φ1 ⋃ φ2) = Sat(∃φ1 ⋃ φ2)
5  |  end
6  |  if ∃s ∈ S : Post(s) ∩ Sat(∃φ1 ⋃ φ2) ≠ ∅ then
7  |  |  foreach s ∈ S : Post(s) ∩ Sat(∃φ1 ⋃ φ2) ≠ ∅ do
8  |  |  |  if s ∈ Sat(φ1) then
9  |  |  |  |  Sat^{t+1}(∃φ1 ⋃ φ2) ← Sat^t(∃φ1 ⋃ φ2) ∪ {s}
10 |  |  |  end
11 |  |  end
12 |  end
13 end
14 return Sat(∃φ1 ⋃ φ2)
```

values of t[18]. For $t = 0$, $Sat^t(\exists \phi_1 \bigcup \phi_2) = Sat(\phi_2)$. For each $t \geq 0$:

$$Sat^{t+1}(\lambda) = Sat^t(\lambda) \cup \{s \in Sat(\phi_1) \mid Post(s) \cap Sat^t(\lambda) \neq \emptyset\}$$

The procedure terminates when $Post(s) = \emptyset$, i.e., when the TS reaches a final state. As we assume TS to be finite, this always happens within a finite time-horizon.

Finally, for $\lambda := \forall \psi$, the algorithm simply convert the universal into an existential quantification following the equivalence rules: $\forall \psi \iff \neg \exists \neg \psi$, then it computes the respective subroutine.

2.7 Probabilistic Computation Tree Logic

Classical model-checking techniques consider "absolute" requirements, such as "the system never loses the message". For many contemporary systems, which are inherently stochastic, these requirements are too strong. For those systems, desirable properties typically have a probabilistic form, such as "the probability that the system loses the message

[18]Notice that each layer of the computation tree describes a specific time-step t, starting from the root, which describes $t = 0$, until the leaves, which describe the termination time-step.

is lower than 0.1". Studying stochastic systems and their probabilistic properties is the aim of *probabilistic model-checking*, a relatively new branch of model-checking that has experienced growing development in the last years. Its starting point can be considered the development of the *Probabilistic Computation Tree Logic* (PCTL) by Hansson and Jonsson in 1994 [78]. The latter is an extension of CTL suitable to model probabilistic properties of stochastic systems and whose semantics is based on Markov chains described in Chapter 1. This section is dedicated to present PCTL and the related model-checking procedures. Specifically, section 2.7.1 introduces PCTL syntax, Section 2.7.2 presents its semantics, and Section 2.7.3 its dedicated to model-checking algorithms. Finally, Section 2.7.4 provides a small demonstrative example of a model-checking task based on PCTL.

2.7.1 PCTL Syntax

The PCTL syntax is recursively defined as follows:

$$\phi := \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid P_{\nabla b}\psi\,, \tag{2.3}$$

$$\psi := \bigcirc \phi \mid \phi_1 \bigcup \phi_2 \mid \phi_1 \bigcup^{\leq t} \phi_2\,. \tag{2.4}$$

The language includes both states-formulae (ϕ) and path-formulae (ψ). ϕ-formulae include the standard notation $\top$ for tautologies, atomic formulae p, standard Boolean connectives for negation and disjunction, and a probabilistic modal operator $P_{\nabla b}$ nesting ψ-formulae and whose informal reading is the following:

- $P_{\nabla b}\psi$: "the probability to reach a path that satisfies ψ is less/equal/greater than b".

Notice that ∇ is a short notation for $<, \leq, =, \geq, >$, while b denotes a real value in the norm $[0, 1]$. The weighted-probabilistic modality replaces CTL quantifiers and specifies probabilities over paths satisfying ψ. This operator allows PCTL to be much more expressive than CTL. For example, we can use PCTL to specify desirable properties such as: "the probability that the system loses the message is lower than 0.2" or "the probability that the system correctly predicts x is greater than 0.9" and so on.

Another peculiarity of PCTL language is the introduction of the new path-operator $\phi_1 \bigcup^{\leq t} \phi_2$. This operator models so-called *time-bounded reachability* and its informal reading is the following:

- $\phi_1 \bigcup^{\leq t} \phi_2$ means: "ϕ_1 along the path until time-step t, then ϕ_2 holds"

2.7.2 PCTL Semantics

The standard semantics for PCTL use Markov chains introduced in Chapter 1 to represent stochastic state-transition systems. There exist a strict connection between Markov chains and transition systems, which is made explicit by the following representability theorem.

Theorem 9 (Representability) *Each transition system (without actions)* $M_{\mathsf{TS}} := \langle \mathcal{S}, \rightarrow, I, AP, l \rangle$ *can be represented by a labelled discrete-time Markov chain* M_{DTMC}.

Proof 4 *For the proof, it is sufficient to observe that all the elements of* M_{TS} *can be represented via elements of* M_{DTMC}. *First,* M_{TS} *and* M_{DTMC} *are defined over the same state space* $\mathcal{S}$ *and share both the set of atomic propositions* AP *and the labelling function* l. *For the other elements, the following conditions hold:*

- *Given a pair of states* $(s, s') \in \mathcal{S} \times \mathcal{S}$, $s \rightarrow s'$ *holds if and only if* $T(s, s') > 0$, *i.e., a transition is possible if and only if its probability is strictly greater than* 0
- *For each* $s \in \mathcal{S}$, $s \in I$ *if and only if* $\iota(s) > 0$, *i.e., a state counts as initial if and only if its initial probability is strictly greater than* 0

Furthermore, we say that a state $s \in \mathcal{S}$ *in a Markov chain* M_{DTMC} *is* absorbing *if and only if* $T(s, s') = 0$ *for all* $s' \neq s$. *It is easy to note that absorbing states are the equivalent of terminal states in a transition system. Indeed, if* $T(s, s') = 0$ *for all* $s' \neq s$, *then no transitions from* s *to any* $s' \neq s$ *are allowed and, thus,* s *is a terminal state. Conversely, if no transitions from* s *to any* $s' \neq s$ *are allowed, then* $T(s, s') = 0$ *for all* $s' \neq s$ *and, thus,* s *is an absorbing state.*

This connection allows for defining a semantics for PCTL by introducing specific satisability conditions for PCTL formulae with Markov chains used as models.

Definition 41 (Satisfiability of ϕ formulae) *Given a labelled DTMC* M_{DTMC} *and a state* $s \in \mathcal{S}$, *the following conditions hold:*

$$
\begin{array}{ll}
M_{\mathsf{DTMC}}, s \models \top, & \forall s \in \mathcal{S} \\
M_{\mathsf{DTMC}}, s \models p & \text{iff } p \in l^i(s) \\
M_{\mathsf{DTMC}}, s \models \phi_1 \wedge \phi_2 & \text{iff } M_{\mathsf{DTMC}}, s \models \phi_1 \wedge M, s \models \phi_2 \\
M_{\mathsf{DTMC}}, s \models \neg\phi & \text{iff } M_{DTMC}, s \not\models \phi \\
M_{\mathsf{DTMC}}, s \models P_{\nabla b}\psi & \text{iff } P(s \models \psi)\nabla b\,,
\end{array}
$$

where $P(s \models \psi)$ denotes the probability that a path $\pi \models \psi$ belongs to the set of paths originating in s conditional to $S_0 = s$. The specific methods to compute this probability vary depending on the formula ψ nested under the probabilistic operator, which will be later presented in Section 2.7.3

Definition 42 (Satisfiability of ψ formulae)

$$
\begin{array}{ll}
M_{\mathsf{DTMC}}, \pi \models \bigcirc \phi & \text{iff } M_{\mathsf{DTMC}}, \pi(1) \models \phi \\
M_{\mathsf{DTMC}}, \pi \models \phi_1 \bigcup^{\leq t} \phi_2 & \text{iff } \exists \tau \leq t : M_{\mathsf{DTMC}}, \pi(\tau) \models \phi_2 \wedge \forall \tau' : 0 \leq \tau' < \tau,\ M_{\mathsf{DTMC}}, \pi(\tau') \models \phi_1 \\
M_{\mathsf{DTMC}}, \pi \models \phi_1 \bigcup \phi_2 & \text{iff } \exists \tau \geq 0 :\ M_{\mathsf{DTMC}}, \pi(\tau) \models \phi_2 \wedge\ \forall \tau' : 0 \leq \tau' < \tau,\ M_{\mathsf{DTMC}}, \pi(\tau') \models \phi_1
\end{array}
$$

A final remark concerns the PCTL expressiveness compared to CTL:

Theorem 10 (PCTL Expressiveness) *PCTL is more expressive than CTL and includes the latter as a special fragment*

Theorem 10 can be proved by means of the following simple equivalence rules that allows to derive CTL quantifiers from the PCTL probabilistic modality:

$$
\begin{array}{rcl}
\forall \psi & \Longleftrightarrow & P_{=1} \psi \\
\exists \psi & \Longleftrightarrow & P_{>0} \psi
\end{array}
$$

Proof 5 *To prove the validity of these rules with respect to the PCTL semantics above introduced we reason as follows. By definition* (40)*, we know that $M_{\mathsf{TS}}, s \models \forall \psi$ if and only if $M_{\mathsf{TS}}, \pi \models \psi$ for all the paths π of M_{TS} such that $\pi(0) = s$. The same condition holds also for the labelled DTMC M_{DTMC} that represents TS, as granted by Theorem* (9)*. Consequently, $M_{\mathsf{DTMC}}, s \models \forall \psi$ if and only if $M_{\mathsf{DTMC}}, \pi \models \psi$ for all the paths π of M_{DTMC} such that $\pi(0) = s$. On the other hand, by Definition 41, we know that $M, s \models P_{=1} \psi$ if and only if $P(s \models \psi) = 1$. Since $P(s \models \psi)$ is defined as the probability that a path $\pi : \pi(0) = s$ satisfies ψ, we have that $P(s \models \psi) = 1$ if and only if $M, \pi \models \psi$ for all the paths π in M_{DTMC} such that $\pi(0) = s$. Hence:*

$$
\begin{array}{lll}
M_{\mathsf{TS}}, s \models \psi & \text{iff} & \forall \pi . \pi(0) = s :\ M_{\mathsf{TS}}, \pi \models \psi\,, \\
& \text{iff} & \forall \pi . \pi(0) = s :\ M_{\mathsf{DTMC}}, \pi \models \psi\,, \\
& \text{iff} & P(s \models \psi) = 1\,, \\
& \text{iff} & M_{\mathsf{DTMC}}, s \models P_{=1} \psi\,.
\end{array}
$$

An analogous reasoning holds for CTL existential quantified formulae $\exists \psi$.

2.7.3 PCTL Model Checking

Model-checking procedures for PCTL formulae are based on a slightly modified version of the *parsing-tree* algorithm already introduced in Section 2.6.2. Given a labelled DTMC

M_{DTMC} and PCTL state-formula ϕ[19], the algorithm calculates $Sat(\lambda)$ for each sub-formula λ of ϕ, hence it calculates $Sat(\phi)$ by composition. The iterative scheme for computing $Sat(\lambda)$ are the same of standard CTL presented in the previous section, while for probabilistic formulae $P_{\nabla b}\psi$ we apply the schema in Equation (4.4.1).

$$Sat(P_{\nabla b}\psi) := \{s \in \mathcal{S} : P(s \models \psi)\nabla b\}\,. \tag{2.5}$$

The specific procedures to compute $P(s \models \psi)$ are based on the feasible inferences on DTMCs described in Chapter 1, notably *marginalization* and *hitting probability*. They vary depending on ψ:

- When $\psi := \bigcirc\phi$, $P(s \models \psi)$ is computed as the *marginal probability* of the event $Sat(\phi)$ given $s \in \mathcal{S}$ be the initial state:

$$P(s \models \bigcirc\phi) := \sum_{s' \in Sat(\phi)} T(s, s') \tag{2.6}$$

- When $\psi := \phi_1 \bigcup^{\leq t} \phi_2$, $P(i, s \models \psi)$ corresponds to $(\mathbf{h}^i)^{\leq t}_{Sat(\phi_2)}(s)$, which is the time-bounded hitting probability of the event $Sat(\phi_2)$ computed through the modified transition matrix $\mathbf{T}^i$ that is obtained from T^i by making all the states $s' \in \mathcal{S} \setminus \{Sat_i(\phi_1) \setminus Sat_i(\phi_2)\}$ absorbing. Accordingly, the algorithm computes $(\mathbf{h}^i)^{\leq t}_{Sat_i(\phi_2)}$ by first generating the modified transition matrix $\mathbf{T}^i$ and then solving Equation (2.7).

$$(\mathbf{h}^i)^{\leq t}_{Sat_i(\phi_2)}(s) := \sum_{s' \in Sat_i(\phi_2)} (\mathbf{T}^i)^t(s, s')\,. \tag{2.7}$$

- When $\psi := \phi_1 \bigcup \phi_2$, $P(s \models \psi)$ corresponds to a particular kind of *hitting probability*, that is, the hitting probability of the conditional event $Sat(\phi_2) \mid Sat(\phi_1)$ $h_{Sat(\phi_2)|Sat(\phi_1)}(s)$, defined as the probability of reaching a state $s' \in Sat(\phi_2)$ with the additional condition that all the states visited before reaching an $s' \in Sat(\phi_2)$ are in $Sat(\phi_1)$. To compute this probability, we introduce a slightly modified version of the general schema for hitting probability presented in (1.8), where $h_{Sat(\phi_2)|Sat(\phi_1)}(s)$ is given by the minimal[20] non-negative solutions of the following system of linear equations:

$$h_{Sat(\phi_2)|Sat(\phi_1)}(s) := \begin{cases} 1 \;\text{ if }\; s \in Sat(\phi_2)\,, \\ 0 \;\text{ if }\; s \notin Sat(\phi_1)\,, \\ \sum_{s' \in Sat(\phi_1)} T(s, s') \cdot h_{Sat(\phi_2)|Sat(\phi_1)}(s') \;\text{ otherwise}\,. \end{cases} \tag{2.8}$$

[19] Notice that, as for CTL, path-formulae ψ are usually not considered in a typical probabilistic model-checking workflow, see [10].

[20] Here, minimality is defined as for Equation (1.8).

Compared to the the system in Equation (1.8), the system in Equation (2.8) includes an additional constraint that, at each $t \in \mathbb{N}$, forces the algorithm to iterate the recursion only over outgoing paths π such that $\pi(t+1) \models \phi_1$. In such a way, only paths π that satify ϕ_1 in all states prior to the first one satisfying ϕ_2 are considered, as required by the condition expressed by the formula $\phi_1 \bigcup \phi_2$.

Notice that, when $\phi_1 = \top$ the schema in Equation (2.8) collapses on the schema in Equation (1.8). This is an expected result. In fact, when $\psi = \top \bigcup \phi_2$, $P(s \models \top \bigcup \phi_2) = h_{Sat(\phi_2)|\mathcal{S}}$, where the latter is the hitting probability of $Sat(\phi_2)$ with the additional condition that all states visited before reaching $Sat(\phi_2)$ are in $\mathcal{S}$. As $\mathcal{S}$ includes all the possible states, $h_{Sat(\phi_2)|\mathcal{S}}$ is in fact equivalent to $h_{Sat(\phi_2)}$.

2.7.4 Example of a PCTL Application

Example 4 (Communication Channel) We now present a simple example of model-checking based on PCTL. The example is borrowed from [10, p.739] Consider a simple communication model operating with a single channel. The channel is error-prone, meaning that messages can be lost. In this particular example, the (four) states of such model M_{DTMC} are in one-to-one correspondence with the atomic propositions, i.e., $\mathcal{S} = AP := \{\text{start}, \text{try}, \text{lost}, \text{delivered}\}$. Transition probabilities are shown in Figure 2.3 as labelled arrows, impossible transitions correspond to missing edges.

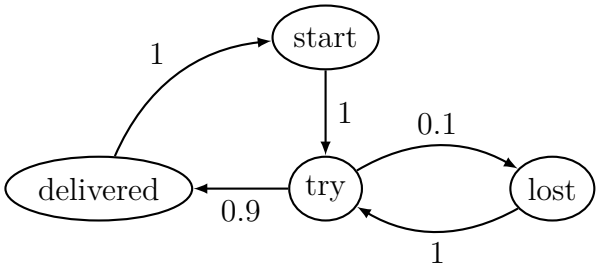


Figure 2.3: A Markov-chain model of the communication channel.

M_{DTMC} is compliant if and only if "*the probability of a message to be lost within seven time steps is smaller than or equal to* 0.25". As the starting state for M_{DTMC} is by construction $s = \text{start}$, this condition is satisfied if and only if:

$$M_{\mathsf{DTMC}}, \{\text{start}\} \models P_{\leq 0.25} \top \overset{\leq 7}{\bigcup} (\{\text{lost}\}) \,, \tag{2.9}$$

Yet, the task reduces to compute $h^{\leq 7}_{\{\text{lost}\}}(\mathcal{S}|\{\text{start}\})$. Its computation can be achieved either by the recursion in Equation (2.8) leading to the numerical values in Table 2.1. Since the

corresponding values is $h^{\leq 7}_{\{\text{lost}\}}(\text{start}) = 0.190 \leq 0.25$, the system satisfies the requirement and can be considered compliant.

	s			
t	start	deliv.	try	lost
0	0.000	0.000	0.000	1.000
1	0.000	0.000	0.100	1.000
2	0.100	0.000	0.100	1.000
3	0.100	0.100	0.100	1.000
4	0.100	0.100	0.190	1.000
5	0.190	0.100	0.190	1.000
6	0.190	0.190	0.190	1.000
7	**0.190**	0.190	0.271	1.000

Table 2.1

2.8 Multi-Agent Systems

So far, we focused on computational systems composed by one single-agent. In this section, we consider model checking formalisms suitable for multi agent systems (MAS) developed in recent years. The most famous example is arguably represented by the formalism of *interpreted systems* (IS) [167, Ch. 8] and the related *Computation Tree Logic of Knowledge* (CTLK) [113]. This logic extends standard CTL with specific epistemic operators for both single-agent and group knowledge. Other notable examples include the *Computationally Grounded Weighted Doxastic Logic* (COGWED) introduced in [28] and the *Probabilistic Computation Tree Logic of Knowledge* (PCTLK) developed in [175]. The former extends CTLK with a weighted doxastic operator to specify multi-agents' degrees of belief, while the latter extends PCTL with single and multi-agent epistemic modalities to specify both epistemic and probabilistic properties of stochastic multi-agent systems. Both COGWED and PCTLK base their semantics on *Probabilistic Interpreted Systems* (PIS), a class of structures merging interpreted systems with labelled Markov chains described in Section 2.7.2. The rest of this section provides an overview of these tree formalisms.

2.9 CTLK

The language of CTLK is obtained by extending standard CTL (Section 2.6) with specific operators to model epistemic properties of agents in a multi-agent system.

Let $\mathcal{A}$ denote a finite non-empty set of agents $\mathcal{A}$. Notation $i \in \mathcal{A}$ is used to denote single agents while group of agents are denoted by $\Gamma \subseteq \mathcal{A}$. The syntax of CTLK is defined as follows.

Definition 43 (CTLK syntax)

$$\phi := \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \exists\psi \mid \forall\psi \mid K^i\phi \mid E^\Gamma\phi \mid C^\Gamma\phi \mid D^\Gamma\phi$$
$$\psi := \bigcirc\phi \mid \phi_1 \bigcup \phi_2$$

The language includes both state-formulae ϕ and path-formulae ψ. The latter are defined as in standard CTL (see, 39). The former include the meta-variable p for atomic propositions, Boolean negation and conjunction, the CTL existential and universal path-quantification, and four epistemic operators nesting ϕ-formulae and whose informal reading is the following:

- $K^i\phi$: "agent i knows ϕ";
- $E^\Gamma\phi$: "everybody in the group of agents Γ knows ϕ";
- $C^\Gamma\phi$: " it is common knowledge in the group of agents Γ that ϕ";
- $D^\Gamma\phi$: "it is distributed knowledge in the group of agents Γ that ϕ".

K^i denotes the standard epistemic modality to represent knowledge of an agent $i \in \mathcal{A}$, while E^Γ, C^Γ, and D^Γ denotes specific modalities that represent different typologies of group knowledge in $\Gamma \subseteq \mathcal{A}$ informally known as, respectively, *everybody knows*, *common knowledge*, and *distributed knowledge*.

2.9.1 CTLK Semantics

The semantics of CTLK is based on the formalism of *interpreted systems*, which are considered the reference frame for representing state-transition multi-agent systems. In an IS, the possible configurations of each agent $i \in \mathcal{A}$ are described by a finite non-empty set of *local* states $\mathcal{S}^i$, whose elements are denoted by $s^i \in \mathcal{S}^i$. The possible configurations of the whole multi-agent system, on the other hand, are described by a finite non-empty set of *global* states $\mathcal{S}$. The latter is obtained as the Cartesian product of the respective sets of *local* states of all the agents $i \in \mathcal{A}$, i.e., $\mathcal{S} := \times_{i\in\mathcal{A}}\mathcal{S}^i$, where each $s \in \mathcal{S}$ is defined as a tuple $\langle s^i, s^j, \ldots, s^n\rangle$ of $\mid \mathcal{A} \mid$ local states s^i, each one describing the local configuration of the agent $i \in \mathcal{A}$ relative to the global configuration s.

Definition 44 (Interpreted systems) *An IS M_{IS} is a tuple $\langle \mathcal{S}, \mathcal{A}, \rightarrow, AP, l\rangle$ where:*

- $\mathcal{S}$ *is a finite non-empty set of global states;*
- $\mathcal{A}$ *is a finite non-empty set of agents;*
- $\rightarrow \subseteq \mathcal{S} \times \mathcal{S}$ *is a transition relation such that, for each* $s, s' \in \mathcal{S} \times \mathcal{S}$*, the transition* $s \rightarrow s'$ *occurs if and only if all agents* $i \in \mathcal{A}$ *transits from their respective local state* $s_i \in s$ *to their respective local state* $s'_i \in s'$*;*
- AP *is a set of atomic propositions;*
- $l : \mathcal{S} \rightarrow 2^{AP}$ *is a labelling function.*

For each agent $i \in \mathcal{A}$, an *epistemic equivalence relation* (EER) $\sim^i \subseteq \mathcal{S} \times \mathcal{S}$ is introduced such that $s \sim^i s'$ holds if and only if $s^i = s'^i$. In practice, we say that two global states s, s' are *epistemically equivalent* (or alternatively, *epistemically indistinguishable*) for an agent $i \in \mathcal{A}$ if and only if they are identical as far as the agent knows.

Each EER $\sim^i$ induces a partition over $\mathcal{S}$ that we denote by $Eq^{\sim^i}$. Each element $eq^{\sim^i}$ of $Eq^{\sim^i}$ denote an *epistemic equivalence class* (EEC), i.e., a set of global states that are epistemically indistinguishable among each others by the agent $i \in \mathcal{A}$. Furthermore, given a group of agents $\Gamma \subseteq \mathcal{A}$, specific EERs for the different kinds of group knowledge above introduced can be defined as follows:

- *Everybody Knows*: $\sim^\Gamma_E := \bigcup_{\forall i \in \Gamma} \sim^i$
- *Common Knowledge*: $\sim^\Gamma_C := it(\bigcup_{\forall i \in \Gamma} \sim^i)$, where *it* denotes the *iterative closure*
- *Distributed Knowledge*: $\sim^\Gamma_D := \bigcap_{\forall i \in \Gamma} \sim^i$.

Each EER induces a respective partition that we denote, respectively, by $Eq^{\sim^\Gamma_E}$, $Eq^{\sim^\Gamma_C}$, and $Eq^{\sim^\Gamma_D}$.

Definition 45 (Satisfiability conditions (IS)) *Given an IS* M_{IS} *and a global state* $s \in \mathcal{S}$*, the following satisfiability conditions hold:*

$$
\begin{array}{ll}
M_{\mathsf{IS}}, s \models p & \text{iff } p \in l(s)\,, \\
M_{\mathsf{IS}}, s \models \neg\phi & \text{iff } \mathcal{M}_{\mathsf{IS}}, s \not\models \phi\,, \\
M_{\mathsf{IS}}, s \models \phi_1 \wedge \phi_2 & \text{iff } M_{\mathsf{IS}}, s \models \phi_1 \wedge M_{\mathsf{IS}}, s \models \phi_2\,, \\
M_{\mathsf{IS}}, s \models \exists\phi & \text{iff } \exists\pi \in \mathrm{Paths}(s) : M_{\mathsf{IS}}, \pi \models \psi, \\
M_{\mathsf{IS}}, s \models \forall\phi & \text{iff } \forall\pi \in \mathrm{Paths}(s) : M_{\mathsf{IS}}, \pi \models \psi, \\
M_{\mathsf{IS}}, \pi \models \bigcirc\phi & \text{iff } M_{\mathsf{IS}}, \pi(1) \models \phi\,, \\
M_{\mathsf{IS}}, \pi \models \phi_1 \bigcup^{\leq t} \phi_2 & \text{iff } \exists\tau \leq t : \begin{array}{l} M_{\mathsf{IS}}, \pi(\tau) \models \phi_2 \wedge \\ \forall\tau' : 0 \leq \tau' < \tau : M_{\mathsf{IS}}, \pi(\tau) \models \phi_1\,, \end{array} \\
M_{\mathsf{IS}}, \pi \models \phi_1 \bigcup \phi_2 & \text{iff } \exists\tau \geq 0 : \begin{array}{l} M_{\mathsf{IS}}, \pi(\tau) \models \phi_2 \wedge \\ \forall\tau' : 0 \leq \tau' < \tau \; M_{\mathsf{IS}}, \pi(\tau') \models \phi_1\,, \end{array} \\
M_{\mathsf{IS}}, s \models K^i\phi & \text{iff } \forall s', s \sim^i s' : s' \models \phi\,, \\
M_{\mathsf{IS}}, s \models E^\Gamma\phi & \text{iff } \forall s', s \sim_E^\Gamma s' : s' \models \phi\,, \\
M_{\mathsf{IS}}, s \models C^\Gamma\phi & \text{iff } \forall s', s \sim_C^\Gamma s' : s' \models \phi\,, \\
M_{\mathsf{IS}}, s \models D^\Gamma\phi & \text{iff } \forall s', s \sim_D^\Gamma s' : s' \models \phi\,.
\end{array}
$$

For ϕ and ψ-formulae, the satisfiability conditions are defined as in standard CTL (see, Section 2.6.1). For epistemic formulae, the satisfiability conditions are based on epistemic equivalence relations as in standard epistemic logic of interpreted systems (see, [167, Ch. 4]). Specifically, an agent i knows a certain proposition ϕ if and only if ϕ is true in all (global) states $s' \in \mathcal{S}$ that are epistemically equivalent to the actual state $s \in \mathcal{S}$. The same condition is adopted also for groups of agents, while the criterion to determine the extension of the respective ECC varies depending on the specific kind of group knowledge considered. For *everybody knows*, the EEC is obtained as the union of all the EECs of the various agents $i \in \Gamma$. For *common knowledge*, the iterative closure of the union is considered, which corresponds to say that ϕ is common knowledge in Γ if and only if all $i \in \Gamma$ know ϕ and know that all the other agents in Γ know ϕ and so on. Finally, for *distributed knowledge*, the EEC is determined by considering the intersection of the EECs of all $i \in \Gamma$. Notice that each partition $Eq^{\sim^i}$ models the information about the whole multi-agent system that is available by a single agent $i \in \mathcal{A}$. The more this partition is subtle, the more information the agent possesses. Therefore, by considering the intersection of the partitions, we in fact merge together the information possessed by all the various $i \in \Gamma$ and eventually consider a partition that is at least as subtle as the most subtle among the various partitions of all the agents in Γ.

2.9.2 CTLK Model Checking

The model-checking problem for CTLK consists of defining a procedure that takes in input an IS M_{IS}, a state $s \in \mathcal{S}$, and a CTLK ϕ and returns either "yes", if $M_{\mathsf{IS}}, s \models \phi$, or "no" otherwise. Here, we consider a procedure based on extending the *parsing-tree*

algorithm introduced in Section 2.6.2 with a specific subroutine to compute $Sat(\lambda)$ for $\lambda := K^i\phi \mid E^\Gamma\phi \mid C^\Gamma\phi \mid D^\Gamma\phi$. The sub-routine is reported in Figure 8.

Algorithm 5: $Sat(\kappa\phi)$ CTLK

Input: A IS M_{IS} and a CTLK epistemic formula $\kappa\phi$
Output: $Sat(\kappa\phi)$

```
1 Sat(κφ) ← {}
2 Compute Sat(φ) := {s ∈ S | M_IS, s ⊨ φ}
3 foreach eq^{∼^κ} ∈ Eq^{∼^κ} do
4 |   if eq^{∼^κ} ⊆ Sat(φ) then
5 |   |   Sat(κφ) ← {eq^{∼^κ}}
6 |   end
7 end
8 return Sat(κφ)
```

Let κ be a short notation for $K^i \mid E^\Gamma \mid C^\Gamma \mid D^\Gamma$, and let $\sim^\kappa$ be a short notation for $\sim^i|\sim^\Gamma_E|\sim^\Gamma_C|\sim^\Gamma_D$. Given an epistemic equivalence relation $\sim^\kappa$, we denote by $Eq^{\sim^\kappa}$ the partition induced on $\mathcal{S}$ by $\sim^\kappa$. Each element of $Eq^{\sim^\kappa}$ is an epistemic equivalence class that we denote by $eq^{\sim^\kappa}$. The algorithm in Figure 8, works as follows.

1. It takes in input an IS M_{IS} and a CTLK formula $\kappa\phi$;
2. It computes $Sat(\phi)$ by recursively calling the respective subroutine;
3. For each $eq^{\sim^\kappa} \in Eq^{\sim^\kappa}$, it checks whether $eq^{\sim^\kappa} \subseteq Sat(\phi)$. If this is the case, then the algorithm adds the whole equivalence class $eq^{\sim^\kappa}$ to $Sat(\kappa\phi)$.

The main advantage of this procedure is that it does not consider single states $s \in \mathcal{S}$ but works directly on epistemic equivalence classes. This strategy drastically reduces the time necessary for the execution of the procedure, which ultimately results polynomial in $\mid \mathcal{S} \mid \cdot n$, with n denoting the nesting-depth of $\kappa\phi$, i.e., the number of nested instances of epistemic operators occurring in the formula[21]. For the details of the proof, see [112, sec. 2.1].

2.10 PCTLK

Another formalism recently proposed to model-check stochastic multi-agent systems against epistemic and probabilistic properties is the PCTLK introduced in [175]. This logic ex-

[21] For example, the formula $\lambda := K^iC^\Gamma p$ has nesting-depth equals to 2.

tends standard PCTL with epistemic operators for representing both single and group knowledge in a multi-agent system.

Definition 46 (PCTLK syntax)

$$\begin{aligned}\phi &:= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \kappa \mid P_{\nabla b}\psi \mid P_{\nabla b}(\kappa)\\ \psi &:= \bigcirc\phi \mid \phi_1 \overset{\leq t}{\bigcup} \phi_2 \mid \phi_1 \bigcup \phi_2\\ \kappa &:= K^i\phi \mid E^\Gamma\phi \mid C^\Gamma\phi \mid D^\Gamma\phi\end{aligned}$$

where $\nabla := <, \leq, =, \geq, >$.

The language includes state-formulae ϕ, path-formulae ψ, and epistemic formulae κ. ϕ-formulae includes atoms p, Boolean connectives for negation and conjunction, and two probabilistic modalities: $P_{\nabla b}\psi$, which expresses probabilistic quantification over paths and has the same informal reading as in standard PCTL (see, Section 2.7), and $P_{\nabla b}\kappa$, which expresses probabilistic quantification over epistemic formulae and whose informal readings varies depending on the nested formula κ as follows:

- $P_{\nabla b}K^i\phi$: "the probability that i knows ϕ is less/equal/greater than b";
- $P_{\nabla b}E^\Gamma\phi$: "the probability that everybody in the group of agents Γ know ϕ is less/equal/greater than b";
- $P_{\nabla b}C^\Gamma\phi$: "the probability that ϕ is common knowledge in the group of agents Γ is less/equal/greater than b";
- $P_{\nabla b}D^\Gamma\phi$: "the probability that ϕ is distributed knowledge in the group of agents Γ is less/equal/greater than b".

ψ-formulae are defined as in standard PCTL (see, Section 2.7.1). Finally, κ formulae include the four epistemic operators for, respectively, single-agent knowledge K^i, everybody knows E^Γ, common knowledge C^Γ, and distributed knowledge D^Γ whose reading is defined as in CTLK (see, Section 2.9).

2.10.1 PCTLK Semantics

The semantics of PCTLK is modelled over *Probabilistic Interpreted Systems* (PIS), a specific class of structures obtained by merging ISs with labelled discrete-time Markov chains.

Definition 47 (PIS) *A PIS* M_{PIS} *is a tuple:*

$$M_{\mathsf{PIS}} := \langle \mathcal{S}, \mathcal{A}, \{T^i\}_{i\in\mathcal{A}}, AP, l(s)\rangle\,;$$

In practice, a PIS is obtained by replacing the transition relation $\rightarrow$ in an IS (see, Definition 44) with:

- a family of transition matrices $\{T^i\}_{i\in\mathcal{A}}$, such that, for each $i \in \mathcal{A}$, T^i provides a compact description the stochastic behaviour across time of the single agent $i \in \mathcal{A}$
- a family of initial probability distributions $\iota^i : \mathcal{S} \mapsto [0, 1]$, such that, for each $i \in \mathcal{A}$, ι^i identifies the possible initial states of the single agent i.

From the individual transition matrix $T^i, i \in \mathcal{A}$, a global transition matrix T_{PIS} is derived that describes the overall stochastic behaviour of the whole multi-agent system. This is generated by computing, for each $s, s' \in \mathcal{S} \times \mathcal{S}$, the *logarithmic pooling* of the transitions:

$$T_{\mathsf{PIS}}(s, s') := \eta \prod_{i\in\mathcal{A}} T^i(s, s'), \tag{2.10}$$

where η is a *normalizing factor* forcing the transitions to satisfy the condition: $(\forall s \in \mathcal{S})\ \sum_{s'\in\mathcal{S}} T_{\mathsf{PIS}}(s, s') = 1$.

Analogously, a *global* initial probability distribution $\iota_{\mathsf{PIS}} : \mathcal{S} \rightarrow [0, 1]$ is derived by computing the logarithmic pooling of the initial distribution of the single agents $i \in \mathcal{A}$:

$$\iota_{\mathsf{PIS}}(s) := \eta' \prod_{i\in\mathcal{A}} \iota^i(s) \tag{2.11}$$

where η' also denotes a normalizing factor that forces the global distribution to satisfy the condition $\sum_{s\in\mathcal{S}} \iota(s) = 1$.

Together, the global transition matrix T_{PIS} and the global initial probability distribution define a DTMC $M_{MC} := \langle \mathcal{S}, \iota_{\mathsf{PIS}}, T_{\mathsf{PIS}}, AP, l\rangle$ that specifies the stochastic behaviour across time of M_{PIS} and that is called the *embedded* Markov chain of the PIS. As we will explain below, this DTMC plays a fundamental role in computing model-checking tasks related to PISs.

Finally, specific epistemic equivalence relations for single-agent knowledge $\sim^i$, everybody knows $\sim^\Gamma_E$, common knowledge $\sim^\Gamma_C$, and distributed knowledge $\sim^\Gamma_D$ are introduced and defined as in standard ISs (see, Section 2.9.1).

Definition 48 (Satisfiability conditions) *Given an PIS M_{PIS} and a global state $s \in \mathcal{S}$, the following conditions hold:*

$$
\begin{array}{lll}
M_{\mathsf{PIS}}, s \models p & \text{iff} & p \in l(s)\,, \\
M_{\mathsf{PIS}}, s \models \neg\phi & \text{iff} & \mathcal{M}_{\mathsf{PIS}}, s \not\models \phi\,, \\
M_{\mathsf{PIS}}, s \models \phi_1 \wedge \phi_2 & \text{iff} & M_{\mathsf{PIS}}, s \models \phi_1 \ \wedge \ M_{\mathsf{IS}}, s \models \phi_2\,, \\
M_{\mathsf{PIS}}, \pi \models \bigcirc\phi & \text{iff} & M_{\mathsf{PIS}}, \pi(1) \models \phi\,, \\
M_{\mathsf{PIS}}, \pi \models \phi_1 \bigcup^{\leq t} \phi_2 & \text{iff} & \exists \tau \leq t : \begin{array}{l} M_{\mathsf{PIS}}, \pi(\tau) \models \phi_2 \wedge \\ \forall \tau' : 0 \leq \tau' < \tau \ : \ M_{\mathsf{PIS}}, \pi(\tau) \models \phi_1\,, \end{array} \\
M_{\mathsf{PIS}}, \pi \models \phi_1 \bigcup \phi_2 & \text{iff} & \exists \tau \geq 0 : \begin{array}{l} M_{\mathsf{PIS}}, \pi(\tau) \models \phi_2 \wedge \\ \forall \tau' : 0 \leq \tau' < \tau \ M_{\mathsf{IS}}, \pi(\tau') \models \phi_1\,, \end{array} \\
M_{\mathsf{PIS}}, s \models P_{\nabla b}\psi & \text{iff} & P_{PIS}(s \models \psi)\nabla b\,, \\
M_{\mathsf{PIS}}, s \models P_{\nabla b}\kappa & \text{iff} & P_{PIS}(s \models \kappa)\nabla b\,, \\
M_{\mathsf{PIS}}, s \models K^i\phi & \text{iff} & \forall s', s \sim^i s' : s' \models \phi\,, \\
M_{\mathsf{PIS}}, s \models E^\Gamma\phi & \text{iff} & \forall s', s \sim_E^\Gamma s' : s' \models \phi\,, \\
M_{\mathsf{PIS}}, s \models C^\Gamma\phi & \text{iff} & \forall s', s \sim_C^\Gamma s' : s' \models \phi\,, \\
M_{\mathsf{PIS}}, s \models D^\Gamma\phi & \text{iff} & \forall s', s \sim_D^\Gamma s' : s' \models \phi\,.
\end{array}
$$

The satisfiability conditions for PCTL-like ϕ and ψ formulae of PCTLK are analogous to those of standard PCTL (see, Section 2.7.2). The satisfiability conditions for epistemic formulae are based on epistemic equivalence relations and defined as in CTLK (see, Section 2.9.1). The main novelty is represented by formulae composed by nesting the probabilistic operator $P_{\nabla b}$ on epistemic formulae κ. The satisfiability condition for these formulae is based on the probability $P_{\mathsf{PIS}}(s \models \kappa)$. Let $eq^{\sim^\kappa}(s)$ denote the EEC in the partition $Eq^{\sim^\kappa}$ that includes the state $s \in \mathcal{S}$, then $P_{\mathsf{PIS}}(s \models \kappa)$ is defined as follows.

$$P_{\mathsf{PIS}}(s \models K^i\phi) := \frac{\mid eq(s)^{\sim^i} \cap Sat(\phi) \mid}{\mid eq(s)^{\sim^i} \mid}\,, \tag{2.12}$$

$$P_{\mathsf{PIS}}(s \models E^\Gamma\phi) := \frac{\mid eq^{\sim_E^\Gamma}(s) \cap Sat(\phi) \mid}{\mid eq^{\sim_E^\Gamma}(s) \mid}\,, \tag{2.13}$$

$$P_{\mathsf{PIS}}(s \models C^\Gamma\phi) := \frac{\mid eq^{\sim_C^\Gamma}(s) \cap Sat(\phi) \mid}{\mid eq^{\sim_C^\Gamma}(s) \mid}\,, \tag{2.14}$$

$$P_{\mathsf{PIS}}(s \models D^\Gamma\phi) := \frac{\mid eq^{\sim_D^\Gamma}(s) \cap Sat(\phi) \mid}{\mid eq^{\sim_D^\Gamma}(s) \mid}\,. \tag{2.15}$$

Following the classical interpretation of probability discussed in Section 1.2, this semantics computes the probability $P_{\mathsf{PIS}}(s \models \kappa)$ as the ratio between: (i) the number of cases favorable to the event $K^i(\phi)$ (respectively, $E^\Gamma\phi$, $C^\Gamma\phi$, $D^\Gamma\phi$), which corresponds to the size of the set of all states s in the respective EEC that satisfies ϕ, and (ii) the total number of possible cases, which corresponds to the size of the respective EEC.

Algorithm 6: $Sat(P_{\nabla b}\kappa)$

Input: A PIS M_{PIS} and a PCTLK epistemic formula $P_{\nabla b}\kappa\phi$
Output: $Sat(P_{\nabla b}\kappa\phi)$

```
Sat(P_{∇b}κφ) ← {}
Compute Sat(φ) := {s ∈ S | M_IS, s ⊨ φ}
Compute the partition Eq(∼^κ)
foreach eq^{∼^κ} ∈ Eq^{∼^κ} do
    if (|eq^{∼^κ} ∩ Sat(φ)| / |eq^{∼^κ}|) ∇b then
        Sat(P_{∇b}κφ) ← {eq^{∼^κ}}
    end
end
return Sat(P_{∇b}κφ)
```

2.10.2 PCTLK Model Checking

The model-checking problem for PCTLK consists of defining a procedure that takes in input a PIS M_{PIS}, a state $s \in \mathcal{S}$, and a PCTLK formula ϕ and returns either "yes", if $M_{\mathsf{PIS}}, s \models \phi$, or "no" otherwise. Also in this case, the overall procedure is based on the parsing-tree algorithm introduced in Section 2.6.2. The specific procedures to compute $Sat(\lambda)$ when λ is a PCTL-like formula are analogous to those of standard PCTL presented in Section 2.7.3. The specific procedures to compute $Sat(\lambda)$ when λ is an epistemic formula are analogous to those relative to epistemic formulae of CTLK presented in Section 2.9.2. The main novelty introduced by [175] consists of a specific sub-routine to compute $Sat(\lambda)$ when $\lambda := P_{\nabla b}\kappa$[22], which is reported in Figure 6.

The algorithm in Figure 6 works as follows. First, it takes in input a PIS M_{PIS} and a PCTLK formula $P_{\nabla b}\kappa\phi$, then it proceeds via the following steps:

1. it selects the proper sub-routine depending on the specification of ϕ and computes $Sat(\phi)$;

2. it calculates the ratio $\frac{|eq^{\sim^\kappa} \cap Sat(\phi)|}{|Eq^{\sim^\kappa}|}$ and checks whether it satisfies the specified threshold ∇b;

3. If the ratio satisfies the threshold, then the algorithm adds the whole equivalence class $eq^{\sim^\kappa}$ to $Sat(P_{\nabla b}\kappa\phi)$.

In practice, the only difference between the PCTLK sub-routine in Figure 6 and the CTLK subroutine reported in Figure 6 lies in the checking step. In the former case, this

[22] Remember that $\kappa := K^i \mid E^\Gamma \mid C^\Gamma \mid D^\Gamma$.

consists of checking whether $eq^{\sim^\kappa} \subseteq Sat(\phi)$, while in the latter case it consists of checking whether the ratio $\frac{|eq^{\sim^\kappa} \cap Sat(\phi)|}{|Eq^{\sim^\kappa}|} \nabla b$ satisfies the specified threshold. This slight modification does not alter the overall time-complexity of the procedure, which remains polynomial in $| \mathcal{S} | \cdot n$.

Notice that an alternative procedure for PCTLK model checking is also presented in [175]. The latter consists of reducing the model-checking tasks for PCTLK to equivalent procedures computable in PRISM [100]. For more details on this alternative procedure, we refer to [175, p.287-293].

2.11 COGWED

The other formalism relevant to model-check multi-agent systems against epistemic properties is the Computationally-grounded Weighted Doxastic (COGWED) logic introduced in [28] and specifically conceived to specify *degrees of belief* in a system of agent.

Definition 49 (COGWED syntax)

$$\phi := p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \exists\phi \mid K^i\phi \mid E^\Gamma\phi \mid C^\Gamma\phi \mid D^\Gamma\phi$$
$$\psi := \bigcirc\phi \mid G\phi \mid \phi_1 \bigcup \phi_2$$

The language of COGWED includes both state-formulae ϕ and path-formulae ψ. ϕ-formulae include atoms p, Boolean connectives for negation and conjunction, existential and universal path-quantifiers, epistemic operators for single-agent knowledge, everybody knows, common and distributed knowledge, and a *weighted doxastic operator* $B^i_{\nabla b}$ nesting ϕ-formulae and having the following informal reading:

- $B^\Gamma_{\nabla b}\phi$: "the degree of belief in ϕ of the group of agents Γ is ∇b"

ψ-formulae includes the standard *next* $\bigcirc$ and *until* $\bigcup$ operator and the operator $G\phi$ (called, the *globally* operator) whose informal reading is the following:

- $G\phi$ means: "ϕ holds globally along the path".

Notice that the latter is sometimes included also in the language of standard CTL and PCTL. However, [10] omits it from both the treatment of CTL and PCTL. Since [10] represents our reference treatise in writing this thesis, we also decided to omit the reference to the globally operator in the previous discussion of CTL and PCTL. In [28], this operator is explicitly mentioned in the syntax of COGWED and, for coherence, we include it in the language.

2.11.1 COGWED Semantics

The COGWED logic is provided with two different semantics, one based on interpreted systems (IS) introduced in Section 2.9.1, and the other based on probabilistic interpreted systems (PIS) introduced in Section 2.10.1. In what follows, both of them are briefly presented and discussed.

Interpreted Systems Semantics.

The IS-based semantics of COGWED is obtained by extending the analogous semantics of CTLK introduced in Section 2.9.1 with two additional satisfiability conditions for the *globally* and the *weighted-doxastic* operators.

Definition 50 (Satisfiability condition (IS)) *Given an IS M_{IS} and a global state $s \in \mathcal{S}$, the following satisfiability conditions hold:*

$$M_{\mathsf{IS}}, \pi \models G\phi \text{ iff } \forall t \geq 0,\ \pi(t) \models \phi\,,$$
$$M_{\mathsf{IS}}, s \models B^{\Gamma}_{\nabla b}\phi \text{ iff } \frac{\mid Sat(\phi) \cap eq^{\sim^{\Gamma}_{D}}(s) \mid}{\mid eq^{\sim^{\Gamma}_{D}}(s) \mid}\ \nabla b\,,$$

where $Sat(\phi) := \{s \in \mathcal{S} \mid M_{\mathsf{IS}}, s \models \phi\}$.

As can be noticed, the *degree of belief* is here interpreted as a *classical probability*, i.e., as a ratio between favorable and possible cases, where, in this case, the "favorable cases" are the states in $eq^{\sim^{\Gamma}_{D}}(s)$ that satisfy ϕ while the "possible cases" are all the states in $eq^{\sim^{\Gamma}_{D}}(s)$. This implies some limitations. As we explained in Chapter 1, the classical interpretation of probability is bound by the assumption that all possible cases are equally probable. In this specific context, this is equivalent to assuming that the probability of reaching an $s' \in eq^{\sim^{\Gamma}_{D}}(s)$ from the actual state $s \in \mathcal{S}$ is the same for all $s' \in eq^{\sim^{\Gamma}_{D}}(s)$. In many real-world scenarios, this assumption results quite unrealistic and limits the applicability of the formalism. For this reason, a different, "more realistic", semantics is presented in [28] based the formalism of *probabilistic interpreted systems* previously introduced.

Probabilistic Interpreted Systems Semantics.

The PIS semantics relies on the notion of *steady-state probability* for an event $B \subseteq \mathcal{S}$, i.e., the probability that the process eventually visit a state in B starting from one if its initial state. Consider the notion of hitting probability h_B of an event B introduced in Chapter 1 for DTMCs. An analogous definition of hitting probability $h_B^{M_{\mathsf{PIS}}}$ for a PIS can be defined, where the latter is simply the hitting probability of B computed computed on the *embedded*

DTMC of the PIS. The steady-state probability distribution of B can thus be obtained simply by multiplying the initial probability vector $\iota^{M_{\mathsf{PIS}}}$ for the hitting probability vector $h_B^{M_{\mathsf{PIS}}}$. That is, the steady-state probability is nothing but the "unconditional" version of hitting probability.

To introduce the PIS-based COGWED semantics, let now us to consider the events $Sat(\phi) \cap eq^{\sim_D^\Gamma}(s)$ and $eq^{\sim_D^\Gamma}(s)$. Their respective steady-state probabilities are given by $\iota^{M_{\mathsf{PIS}}} h^{M_{\mathsf{PIS}}}_{Sat(\phi)\cap eq^{\sim_D^\Gamma}(s)}$ and $\iota^{M_{\mathsf{PIS}}}(s) h^{M_{\mathsf{PIS}}}_{eq^{\sim_D^\Gamma}(s)}$. The latter corresponds to the probability of eventually visiting a state in the equivalence class, while the former corresponds to the probability of eventually reaching a state in the equivalence class *that also satisfies* ϕ. The degree of belief is thus interpreted as the ratio between these two steady-state probabilities, as reported in Definition 51.

Definition 51 (Satisfiability conditions (PIS)) *Given a PIS M_{PIS}, a state $s \in \mathcal{S}$, and a formula $B^\Gamma_{\nabla b}\phi$, the following condition holds:*

$$M_{\mathsf{PIS}}, s \models B^\Gamma_{\nabla b} \;\text{ iff }\; \frac{\iota^{M_{\mathsf{PIS}}}(s) h^{M_{\mathsf{PIS}}}_{Sat(\phi)\cap eq^{\sim_D^\Gamma}(s)}(s)}{\iota^{M_{\mathsf{PIS}}}(s) h^{M_{\mathsf{PIS}}}_{eq^{\sim_D^\Gamma}(s)}(s)} \nabla b\,.$$

Differently from the counting-world semantics, this new semantics thus takes the stochastic law (i.e., the probability distribution) governing the whole multi-agent system evolution on the long-term into account. In this regard, hence, it is no longer bound to the unrealistic assumption that all the states in $eq^{\sim_D^\Gamma}(s)$ are equally probable in terms of steady-state probability.

2.11.2 COGWED Model Checking

The model-checking problem for COGWED consists of defining a procedure that takes in input either a IS M_{IS} or a PIS M_{PIS}, a state $s \in \mathcal{S}$, and a state-formula ϕ and returns "yes" if $M_{\mathsf{IS}}, s \models \phi$ ($M_{\mathsf{PIS}}, s \models \phi$) and "no" otherwise. Also in this case, the overall procedure relies on the *parsing-tree* introduced in Section 2.6.2. The procedure to compute $Sat(\lambda)$ when $\lambda = K^i\phi \mid E^\Gamma\phi \mid C^\Gamma\phi \mid D^\Gamma\phi$ is identical to that introduced in Section 2.9.2 for analogous CTLK epistemic formulae. Two specific sub-routines are introduced to compute $Sat(\lambda)$ when λ is a weighted-belief formula. The former, based on IS-semantics, is reported in Figure 7. The latter, based on PIS-semantics, is reported in Figure 8.

Algorithm of IS-Semantics Subroutine.

Algorithm 7: $Sat(B^{\Gamma}_{\nabla b}\phi)$

Input: A PIS M_{PIS} and a COGWED weighted-doxastic formula $B_{\nabla b}\phi$
Output: $Sat(B^{\Gamma}_{\nabla b}\phi)$

$Sat(B^{\Gamma}_{\nabla b}\phi) \leftarrow \{\}$
Compute $Sat(\phi) := \{s \in \mathcal{S} \mid M_{\mathsf{IS}}, s \models \phi\}$
Compute the partition $Eq(\sim^{\kappa})$
foreach $eq(\sim^{\kappa}) \in Eq(\sim^{\kappa})$ **do**
  if $\frac{|eq(\sim^{\kappa}) \cap Sat(\phi)|}{|eq(\sim^{\kappa})|} \nabla b$ **then**
    $Sat(B^{\Gamma}_{\nabla b}\phi) \leftarrow \{eq(\sim^{\kappa})\}$
  end
end
return $Sat(B^{\Gamma}_{\nabla b}\phi)$

The algorithm in Figure 7 works as follows. It takes in input a IS M_{IS} and a formula $B^{\Gamma}_{\nabla b}\phi$, then it proceeds through the following steps:

1. it computes $Sat(\phi)$ by means of the proper procedure;

2. For each $eq^{\sim^{\Gamma}_{D}}(s) \in Eq^{\sim^{\Gamma}_{D}}(s)$, it computes the ratio $\frac{|Sat(\phi) \cap eq^{\sim^{\Gamma}_{D}}(s)|}{|eq^{\sim^{\Gamma}_{D}}(s)|}$ and checks whether it satisfies the specified threshold ∇b;

3. If the ratio satisfies the threshold, then the algorithm adds the whole EEC $eq^{\sim^{\Gamma}_{D}}(s)$ to $Sat(B^{\Gamma}_{\nabla b}\phi)$.

For what concerns the time-complexity of the procedure, it results to be polynomial in $| \mathcal{S} | \cdot n$, with n being the nesting-depth of ϕ, as clarified in [28, p. 6].

Algorithm for PIS-Semantics Subroutine

Algorithm 8: $Sat(B^{\Gamma}_{\nabla b}\phi)$

Input: A PIS M_{IS} and a COGWED weighted-doxastic formula $B_{\nabla b}\phi$
Output: $Sat(B^{\Gamma}_{\nabla b}\phi)$

$Sat(B^{\Gamma}_{\nabla b}\phi) \leftarrow \{\}$
Compute $Sat(\phi) := \{s \in \mathcal{S} \mid M_{\mathsf{IS}}, s \models \phi\}$
Compute the partition $Eq^{\sim^{\Gamma}_{D}}(s)$
foreach $eq^{\sim^{\Gamma}_{D}}(s) \in Eq^{\sim^{\Gamma}_{D}}(s)$ **do**
    Compute $h^{M_{\mathsf{PIS}}}_{Sat(\phi)\cap eq^{\sim^{\Gamma}_{D}(s)}}(s)$ through the schema in Equation (1.8)
    Compute $h^{M_{\mathsf{PIS}}}_{eq^{\sim^{\Gamma}_{D}(s)}}(s)$ through the schema in Equation (1.8)
    if $\frac{\iota^{M_{\mathsf{PIS}}}(s)h^{M_{\mathsf{PIS}}}_{Sat(\phi)\cap eq^{\sim^{\Gamma}_{D}(s)}}(s)}{\iota^{M_{\mathsf{PIS}}}(s)h^{M_{\mathsf{PIS}}}_{eq^{\sim^{\Gamma}_{D}(s)}}(s)} \nabla b$ **then**
        $Sat(B^{\Gamma}_{\nabla b}\phi) \leftarrow \{eq^{\sim^{\Gamma}_{D}}(s)\}$
    end
end
return $Sat(B^{\Gamma}_{\nabla b}\phi)$

The algorithm in Figure 8 works as follows. It takes in input a PIS M_{PIS} and a formula $B^{\Gamma}_{\nabla b}\phi$, then it proceeds through the following steps:

1. it computes $Sat(\phi)$ by means of the proper procedure;
2. it computes both $h^{M_{\mathsf{PIS}}}_{Sat(\phi)\cap eq^{\sim^{\Gamma}_{D}}}(s)$ and $h^{M_{\mathsf{PIS}}}_{eq^{\sim^{\Gamma}_{D}(s)}}$, by means of the procedure reported in Equation (1.8);
3. it compute the ratio $\frac{\iota^{M_{\mathsf{PIS}}}(s)h^{M_{\mathsf{PIS}}}_{Sat(\phi)\cap eq^{\sim^{\Gamma}_{D}(s)}}(s)}{\iota^{M_{\mathsf{PIS}}}(s)h^{M_{\mathsf{PIS}}}_{eq^{\sim^{\Gamma}_{D}(s)}}(s)}$ and checks whether it satisfies the specified threshold ∇b.
4. If the ratio satisfies the threshold, then the algorithm adds the whole EEC $eq(\sim^{\Gamma}_{D})$ to $Sat(B^{\Gamma}_{\nabla b}\phi)$.

For what concerns the time-complexity of the whole procedure, again it results polynomial in $\mid \mathcal{S} \mid \cdot n$ as clarified in [28, p. 6].

Chapter 3

Probabilistic Model Checking for Explainable AI

Abstract

This chapter explores a potential connection between model checking and the emerging field of Explainable Artificial Intelligence (XAI). It is divided into two main parts. The first part analyses the *opacity problem* in contemporary machine learning (ML) from a philosophical perspective. It outlines a general ontological account of machine learning and a taxonomy for the different typologies of opacity that one may encounter in dealing with ML systems. It then provides a general overview of the XAI research program and its various methodologies, with a particular emphasis on post-hoc explanation methods based on surrogate models. The second part of the chapter introduces a model checking framework to verify the reliability of post-hoc explanations provided via surrogate models in terms of three fundamental properties: *transparency*, *accuracy*, and *trustworthiness*. The framework is built upon a multi-agent Markovian semantics that includes different typologies of stochastic and epistemic agents. This semantics is inspired by recent works in the model checking of stochastic multi-agent systems, notably [28] and [175]. The chapter is based on previous results published in [160].

3.1 Introduction

The incredible success of information technology (IT) systems in the last years can be considered mostly a consequence of the advancements in machine learning (ML) technologies. ML is a broad field of research that includes a wide variety of tools and methods to make artificial agents able to extract information, learn knowledge and build models from data on their own [155].

In general, computational systems based on ML (ML systems) possess an impressive inferential power that allows them to analyse large amounts of data and identify patterns

that neither the human eye nor traditional statistical methods would likely ever be able to discover [3]. On the other hand, however, they suffer from the problem of being opaque, or, as they say, "black boxes". Loosely speaking, that a ML system is opaque means that it is difficult for users to know how it works, to interpret its decisions, and to evaluate its behaviour against scientific and ethical norms [179]. In recent years, opacity, or "the black box problem", has caught the attention of many scholars, both from philosophy and computer science. For their part, philosophers have proposed different conceptual analyses of the notion of opacity and debated its epistemological and ethical implications for the human society [23, 179, 79, 118, 56]. Computer scientists and engineers, on the other hand, have focused on developing methods and tools to counter opacity effects and obtain *explainable* AI systems [1, 76]. Their joint efforts have led to the birth of *eXplainable AI* (XAI), a new area of research aimed at rendering ML systems less opaque and more humanly understandable [145].

The present chapter analyses the problem of opacity and explores potential connections between the research area of XAI and probabilistic model-checking. Section 3.3 outlines an ontology of ML systems based on the multi-layer ontology of computational artifacts proposed in [131, ch. 11]. Section 3.3 discusses the opacity problem and outlines a taxonomy of its various forms and dimensions. Section 3.4 gives an overview of the state of the art in XAI, presenting different methods and approaches to explaining opaque ML systems. It thus focuses on a particular class of XAI methods, which are post-hoc explanation methods based on surrogate models [76]. In a nutshell, these methods consist of training a surrogate transparent model that emulates a target ML system either globally or locally on a given outcome and serves as a *post-hoc* explanation of the latter. To convey effective and reliable explanations, it is generally required that surrogate models satisfy some fundamental "explainability properties", which we identify as *opacity/transparency*, *accuracy*, and *trustworhtiness*. Our idea is that such properties can be correctly formalized and verified at an abstract specification level using probabilistic model checking techniques. The remaining sections of the chapter are dedicated to develop this idea more in depth. More specifically, Section 3.5 presents a multi-agent Markov models semantics to formalize explainability properties. The general idea beyond this semantics is that explainability properties are best understood as *quantifiable relational properties* that are instantiated between three different types of agents: (i) the opaque target ML system to be explained, (ii) the surrogate models explaining the target system, and (iii) the users of the target system to whom explanations are directed. Section 3.5.1 presents a logic to specify explainability properties called Ex-PCTL. This logic is obtained as an extension of the well-known PCTL introduced in the previous chapter and includes specific modal operators to specify explainability properties. Section 3.5.2 develops a semantics for Ex-PCTL formulae based on the multi-agent environment introduced in Section 3.5. Section 3.6 develops specific algorithms to model-check surrogate models against explain-

ability properties specified in the Ex-PCTL language. These algorithms extend the CTL parsing-tree algorithm introduced in Section 2.6.2 with specific sub-routines for Ex-PCTL formulae inspired by existing techniques for multi-agent extensions of PCTL discussed in Sections 2.10 and 2.11 of the previous chapter. The chapter ends with Section 3.7 presenting an example of applications of the developed formalism to model-check explainability properties of probabilistic decision-tree classifiers.

3.2 Machine Learning Systems

In this section we provide a general account of the ontology of machine learning systems generalizing the multi-level ontology of computational artifacts introduced in [71, 131].

The ontology of traditional digital computational systems represents a central topic of interest in the philosophy of computer science. In literature, two major accounts for it have been proposed, which we can refer to as the *dual* and the *multi* level ontology [6]. The dual ontology understand computational systems as composed by two fundamental entities, *software* and *hardware*, each one possessing its own distinct ontology. The software corresponds to the abstract sets of operations and rules that enable a system to generate specific outputs from given inputs, while the hardware pertains to the physical structures (such as electric circuits and gates) and actions (like circuit closure and voltage inversion) that effectively carry out the operations described in the software. Although popular, the dual ontology has faced significant criticism in contemporary literature. For instance, [71, 131] argue that the dual ontology is too simplistic for being able to account for the high complexity of computational systems. They therefore propose a refined approach that understands computational systems as complex artifacts whose structure and functioning can be described at five different levels of abstraction (LoA) [71, 131]. These levels are arranged in a hierarchy that ranges from electric circuits, at the bottom level, up to include the intentions of the system's designers:

1. **Functional Specification Level** (FSL). This level provides a general description of the overall I/O behaviour of the system in terms of a function mapping each input to its intended output. This function reflects the "intentions" of the system's designer and specifies in very general terms what the system is supposed to do. For example: to calculate fundamental arithmetic operations, to separate annoying emails from relevant ones, or to predict the probability of a disease based on the analysis of medical records.

2. **Design Specification Level** (DSL). This level provides an abstract description of the system, identifying and explaining the specific functions and operations required to fulfill the intended function. This abstract description can be formulated by means

of dedicated *specification languages* (e.g., ACSL, CASL, Alloy etc.), which cannot be directly executed but require to be translated into executable algorithms via a process called *refinement* [131]. Although all specification languages are conceived to provide very abstract descriptions, the design format they adopt may vary a lot depending on the paradigm they refer to. In algebraic approaches, systems are described using abstract algebraic structures like sets, functions, morphisms etc; in logic-oriented approaches, specifications are provided via logical axioms describing the properties and behaviours that the system is supposed to satisfy; finally, in model-oriented approaches, systems' structure and functioning are described through states-based models, such as transition systems or finite-state automata.

3. **Algorithm Design Level** (ADL). This level translates the specifications formulated at the DSL into executable procedures (algorithms), which consist of sequences of rules and operations that the system must execute in order to accomplish its specification and thus fulfill its intended function. At this level, these procedures are not yet described using executable programming languages but higher-level formalisms, like the standard pseudo-code, see [106].

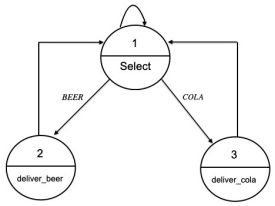
4. **Algorithm Implementation Level** (AIL). This level translates the procedures specified at the ADL into actual executable programs, i.e. sets of instructions in some high-level programming language (such as Python or Java). These are then automatically translated (through compiling and linking) into low-level instructions in Assembly, a programming language with a one-to-one correspondence with the machine code controlling the behaviour of the physical components.

5. **Algorithm Execution Level** (AEL). At this level, corresponding to what is usually called the *hardware*, the instructions provided by the machine code are translated into physical actions performed by electric circuits and gates, hence executed.

Example 5 As an example of "traditional" computational system, let us consider a rudimentary beverage vending machine that offers two options: *beer* and *cola*. The machine operates as follows: when no input is provided by users, the monitor displays the word "select". If a user presses the $BEER$ button, the machine dispenses beer and subsequently reverts to displaying "select". Finally, if a user presses the $COLA$ button, the machine dispenses cola and again returns to displaying "select". At the FSL, the beverage machine can be described via the following intended function f:

$$f := \{\, \langle \emptyset, display\_select \rangle \ \langle BEER, deliver\_beer \rangle \ \langle COLA, deliver\_cola \rangle \,\}$$

The function specifies the *intended behaviour* of the machine in the form of all the possible I/O pairs that are admissible[1]. At the DSL, the machine can be described via a finite state automaton implementing the function f, as in Figure 3.1a. The automaton encompasses three distinct states (1, 2, 3), each one paired with a specific action (*display_select*, *deliver_beer*, *deliver_cola*) that the automaton commands to execute when in that state. Transitions are enabled by the occurrence of specific inputs (no-input, $BEER$, $COLA$). For example, if the automaton is in state 1 and the user presses the button $BEER$, then the automaton transits to state 2 and commands to execute the action *deliver_beer*. Differently, if the automaton is in state 1 and no actions are executed, then it remains in state 1 and continues to execute the action *display_select*. At the ADL, the automaton is implemented in a step-by-step procedure specified using the pseudo-code, as in Figure 3.1b. At the AIL, the pseudo-code is implemented in an executable program, written in some high-level programming language, as in Figure 3.1c. Finally, at the AEL, the program is compiled in machine language and then executed on electric circuits.



(a) Moore automaton

Algorithm 1: The Beverage Machine

Input: *BEER, COLA*
Output: *select, deliver.beer, deliver.cola*

```
if machine in state 1 then
    if no actions occur then
        return select
    end
    if user selects BEER then
        move to state 2 and return deliver.beer
    end
    if user selects COLA then
        move to state 3 and return deliver.cola
    end
end
if machine in state 2 then
    move to state 1 and return select
end
if machine in state 3 then
    move to state 1 and return select
end
```

(b) Abstract algorithm

```python
class BeverageMachine:
    def __init__(self):
        self.state = 1

    def select_beverage(self, beverage):
        if self.state == 1:
            if not beverage:
                return "select"
            if beverage == "BEER":
                self.state = 2
                return "deliver_beer"
            if beverage == "COLA":
                self.state = 3
                return "deliver_cola"
        if self.state == 2 or self.state == 3:
            self.state = 1
            return "select"
```

(c) Python program

Figure 3.1: The Beverage Machine represented at three different LoAs via: (a) a Moore automaton (DSL), (b) an abstract algorithm in pseudo-code (ADL), and (c) a Python program (AIL).

The multi-level ontology, although primarily developed for "traditional" computational systems, can be adapted to some extent to account for the ontology of ML systems. The latter, however, present some peculiarities that have to be taken into account.

One major difference concerns the specification of the system's behaviour at the FSL. In the case of "traditional" computational systems, the intended I/O behaviour is fully specified by the designer. In the case of ML systems, on the other hand, the designer only defines constraints on the I/O behavior, while the specific intended function is *learned* from a set of data through an optimization process called *training*. Therefore, the behaviour of a (trained) ML system is determined not only by its designers' intentions, but also by the information included in the training data. This point is crucial, in particular because information included in datasets might be biased and produce undesired or unfair behaviours that designers cannot ultimately account for.

[1]The notation $\emptyset$ denotes here that no actions are performed.

Another related peculiarity of ML systems concerns their architecture, which is much more complex then that of traditional systems and includes three distinct artifacts interacting with one another to determine the overall behaviour of the system [62]:

- The **training sample**: the data set used to train the system;
- The **training engine**: the algorithm that allows the system to learn from data;
- The **ML model**: the optimized computational model resulting from the process of training.

In what follows, we discuss their nature in some more detail.

Training Sample

The overall I/O behaviour of a ML system is typically determined by a set of *parameters* whose values are learned automatically by the system during the process of training. The *training samples* are the collections of data that a ML system uses to fit such parameters. In this regard, they play a central role in determining the final (i.e., after-training) I/O behaviour of the system.

In a sense, training samples play for ML systems a similar role to what designers' intentions do for traditional computational systems: they provide, albeit implicitly and partially, a specification of their intended behaviour. For this reason we consider them as component-parts of the of ML systems' structure. Although maybe not self-evident, as many might consider a ML system independently from the data on which it has been trained, however, this view is perfectly in line with what subsumed by recent ontological accounts, such as the one proposed in [71, 131]. These latter accounts maintain that the ontology of a computational system has to consider *all* the elements that determine its behaviour: point in case, they typically include the designers' intentions as part of the system itself. Similarly, we argue here that the training sample determines crucially the way in which the algorithm behaves *after training*, and therefore it properly qualifies as an element of its ontology.

The specific nature of training samples varies a lot depending on the application context considered and the training procedure adopted. In so-called *supervised learning*, training samples usually consist of collections of structured data called *examples*. An instance of such structured data would be a pair $\langle X, Y \rangle$ where $X \in \mathcal{X}$ is a compact mathematical representation (vector, matrix, tensor etc.) of data features (e.g., genetic sequences of a genome, or pixels of an image) and $Y \in \mathcal{Y}$ is the desired outcome that we want the system to associate with input X. All components of an example belong to a given domain that is defined by the application task. Consider images classification, where all

Xs will be matrices of pixels and all Ys will be labels for classes of images. Examples are used by the system during training to induce the intended function $f : \mathcal{X} \to \mathcal{Y}$ defining the resulting (after training) I/O behaviour. In practice, what the system does is to set parameters governing its functioning in order to produce an I/O mapping that minimizes the prediction error over examples[2].

In *unsupervised learning*, training samples consist mostly of unstructured collections of data-points whose coordinates encode relevant properties of their target. For example, patients of a medical study can be represented in the training sample through data-points whose coordinates denote their relevant characteristics, such as age, gender, place of birth and so on. Geometrical and mathematical properties of data-points are then analysed during training to find relevant patterns among them. Based on these patterns, the system induces the intended function $f : \mathcal{X} \to \mathcal{Y}$ defining the resulting I/O behaviour.

In *reinforcement learning*, the training sample consists of an interactive environment composed by a state space $\mathcal{S}$ and a reward function R. During the training, the system explores the state space $\mathcal{S}$ getting rewards and punishments depending on its moves, available from a given set $\mathcal{A}ct$. Based on the latter, it induces an intended function $f : \mathcal{S} \to \mathcal{A}ct$, technically called *policy*, that prescribes how the system moves within the state-space $\mathcal{S}$.

In all the case above, training samples may consist of either well-determined datasets sampled from a given population, or ill-defined collections of data from the environment with which the system interacts. In the latter case, we say that the training occurs in an *open environment*. An example is given by AI systems launched on the internet that learn from the interaction with users. Notice that if a system is trained in an open environment, then identifying its borders is practically impossible, although at any given moment of use, the system will have a finite and determinate dataset on which it has been trained. As we assume the training sample to be structurally part of the ML system, this is tantamount to say that ML systems trained in open environments have no static well-defined borders. This claim poses an ontological problem, as it subsumes that the nature of such systems has a potentially always shifting definition of specification, thereby affecting both the intended behaviour and the ability to evaluate correctness in it standard sense. We do not further investigate this aspect here, as it goes beyond the present investigation.

Training Engine

The training engine is the set of procedures that allows a ML system to fit parameters according to the information included in the training sample. Ontologically, it is a computational artifact whose structure and behavior can be described at the various LoAs of

[2]A prediction error over a set of examples $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{Y}$ is a pair $\langle X, Y \rangle \in \mathcal{X} \times \mathcal{Y}$ such that $\exists \langle X, Y' \rangle \in \mathcal{D}$ and $Y \neq Y'$.

the hierarchy presented above.

At the FSL, the intended function of the training engine is to solve an *optimization task*. In mathematics an optimization task is a problem whose solution requires to find the *maximum*, respectively, the *minimum* of a given target function. In the case of ML systems, the specific nature of the target function varies depending on the type of training.

In supervised learning, the target function takes the form of a *loss function* (or *error function*) defined over examples of the target sample. Let W be an array of parameters of a given ML system M and let $f^M_W : \mathcal{X} \to \mathcal{Y}$ be a mapping describing the I/O behaviour of M over the training sample under the assignment of parameters W. A loss function is a function that takes a pair $(Y, f^M_W(X))$ with Y denoting the intended outcome paired with X in the training sample and $f^M_W(X)$ being the outcome associated to X by M under the assignment of parameters W, and returns a real number $L(Y, f^M_W(X)) \in \mathbf{R}$ representing the "predictive error" of the system on input X. The latter is a measure of the "distance" between the actual outcome $f^M_W(X)$ and the desired outcome Y prescribed by the training sample. The optimization task consists in finding an assignment of parameters that minimizes $L(Y, f^M_W(X))$ $\hat{W} = \arg\min_W L(Y, f^M_W(X))$ for each pair $\langle X, Y\rangle$ in the training sample. Accordingly, the *intended function* of the training engine will take the form: $\arg\min_W L(Y, f^M_W(X))$. In practice, only approximate solutions of the optimization task can be achieved through a variety of iterative heuristic methods, such as the well-known *stochastic gradient descent* (see, [19]), whose choice ultimately depends on both the specific ML system adopted and the application context considered.

In unsupervised learning, to provide a general description of the target function is difficult: the latter varies a lot depending on the specific method and task considered. In general, it corresponds to some similarity or dispersion measure (e.g., Jaccard distance, variance etc.) that quantifies how much data-points are related to each other based on their relevant properties. For example, in *K-means* the common target function to be minimized is the variance within a given number of clusters, which measures how far data points in those cluster are spread out from their average value. In *anomaly detection*, on the other hand, it often corresponds to the z-score, which measures how much single data-points are below or above a reference mean value.

In reinforcement learning, the target function is technically called *value function* V^θ and consists of a mapping that assigns to each $s \in \mathcal{S}$ a real-value $V^\theta(s) \in \mathbf{R}$ denoting the expected reward earned by the system starting its exploration of the environment (i.e., the training sample) in s and then following the policy (i.e., the sequence of moves) θ[3]. The optimization task consists here in finding the policy f that maximizes V^θ in the

[3]The reader may note here a strict similarity between this framework and the formalism of Markov decision processes outlined in Chapter 1. This is no coincidence as a reinforcement learning task actually corresponds to the resolution of a Markov decision process. This topic is addressed more in detail in chapter 5, where reinforcement learning is applied to the resolution of a well-known problem in epistemic

long run. Accordingly, the *intended function* of the training engine will take the form: $(\forall s \in \mathcal{S})$ $\arg\max_\theta V^\theta(s)$. Approximate solutions of the optimization task are achieved in practice by a variety of techniques (for more details, see [155]).

At the DSL level, a more detailed description of the optimization task is usually provided, including a specification of the target function to be minimized/maximized and a specification of the heuristic involved to solve the optimization task. Consider for example the training engine of a deep neural network. At the FSL, its intended function can be described simply as that of minimizing a loss function $L(Y, f^M_W(X))$ over examples of the training sample. At the DSL, the designer is required to specify which loss function they intend to adopt, for example the *Mean Squared Error* $\sigma := (Y - f^M_W(X))^2$ or the *Mean Absolute Error* $\hat{m} :=\mid Y - f^M_W(X) \mid$, and which procedure they plan to use to approximate its global minimum, for example *Stochastic Gradient Descent* (see, [19]).

Once the DSL is specified, the construction of the lower LoAs proceeds as in traditional computational systems, that is, by defining an executable procedure to solve the optimization task that takes care of available resources and implementing it in a program that is eventually executed on a physical machine.

Machine Learning Model

The ML model, also called *learned model* or, in case of binary predictors, *learned classifier*, is the optimized computational model that results from the training process and which is responsible for the the I/O behaviour of the trained ML system.

Semantically, a ML model is a *model of data*, i.e., it denotes the statistical regularities among data in a given domain. Epistemologically, on the other hand, it can be regarded as a *predictive* rather than an *explanatory* model, that is, it is useful in predicting new instances of a given target phenomenon by induction from instances in the training sample, while it does not provide explanations as to *why* the target phenomena occur in a certain way. From the ontological point of view, the ML model is a computational artifact which, like the training engine, can be described at the different LoAs of the above hierarchy.

At the FSL, a ML model usually takes the form of a *joint* probability distribution $P(X_1, \ldots, X_n, Y)$ over a set of stochastic variables $X_1, \ldots, X_n$ representing features and a variable Y ranging over possible outcomes of the ML system. The complexity of this distribution is usually defined in terms of:

(i) the number n of stochastic variables it involves, and

(ii) the possibility to decompose it in simpler distributions describing specific statistical dependencies among single pairs of variable X_i, X_j, $i, j \in \mathbf{N}$.

---

logic, that of logical omniscience.

Complexity largely depends on the specific kind of ML model considered. Linear and multiple regression models usually involve a small number of variables and can be easily decomposed into simpler distributions. Bayesian networks may include a relatively high number of variables but can be easily decomposed thanks to their mathematical properties, in particular the *Markov property* stating that $P(X_i|X_1,\ldots,X_n) = P(X_i|\mathrm{Pre}_i)$, where Pre_i denotes the parents of X_i in the Bayesian network's graph. Deep learning models, like transformers or recurrent neural networks, usually involve a very high number of variables and are difficult, if not impossible, to decompose, and for this reason are usually considered among the most complex instances of ML models [92].

At the DSL, the ML model consists of a full specification of the two classes of elements that govern the I/O behaviour of the trained ML system: *hyper-parameters* and *parameters*. Hyper-parameters are structural elements of the system's architecture that constrain its behaviour. Their specification is provided by the designer on the basis of considerations regarding the type of task that the ML system is supposed to perform (e.g., classification, regression) and the specific application domain (e.g., image recognition, natural language processing etc.). Parameters, on the other hand, are the components that determine the specific behaviour of the ML system within the constraints given by the hyper-parameters. Their specification is an automatic task performed by the training engine on the basis of information included in the training sample.

Hence, the peculiarity of ML models as computational artifacts consists in the fact that their designing process is a joint human-machine effort: part of it, namely the specification of the hyper-parameters, is performed by a human designer, whereas the other, i.e., the specification of the parameters, is automatically executed by the training engine. A direct consequence of this fact is that, while in the case of traditional models the specification of their design is fully determined by the designer on the basis of their intended function (e.g., filtering spam), in case of ML models it is determined by two distinct elements: (i) their intended function (e.g., classifying images) defined by the designer and reflected in the choice of the hyper-parameters, and (ii) the information included in the training sample, which is instead reflected in the setting of the parameters.

As for the training engine, once the DSL is defined, the specification of the lower LoAs proceeds as in traditional computational systems, thus taking care of "implementative" issues like tractability and the efficient allocation of computational resources.

3.3 The Opacity Problem

Despite ML systems are nowadays present in many domains of society, there is increasing concern on their use due to the well-known *opacity* or *black box* problem, which roughly arises from the fact that ML systems' behaviours and outcomes are generally hard to

understand and verify [179]. Opacity is actually perceived as a very serious issue, not only for its technical relevance but because it undermines users' *right to an explanation*, as recently mentioned in the latest edition of the EU General Data Protection Regulation [171]. Given its relevance, the opacity problem has become nowadays an hot topic among different categories of scholars, including philosophers of science and ethical experts (see, for instance, [23] and [57]). Its perceived relevance resulted in an increasing popularity of the XAI research program, whose central aim is exactly to define methods for countering opacity and make ML models' more humanly understandable [8, 1] Despite the increasing amount of literature on the opacity problem and the incredible popularity recently reached by the XAI research program, however, what exactly means that ML systems are opaque, or "black boxes" is still far from being clear. Instead, the impression is that the term "opacity" is used vaguely and to refer to a variety of related but different problems. Sometimes, in fact, opacity is supposed to mean that a model's complexity prevents human users from understanding its inner functioning [27] or from grasping the inferences it uses to generate a given prediction [56]. In other cases, it is emphasised that the models' design prevents users from understanding and checking the reliability of the learned patterns [76]. In yet other cases, ML models are considered opaque because they only predict occurrences of phenomena but do not explain the causes, mechanisms or laws that govern them [114]. This vagueness is often a source of confusion in the debate and feeds the impression that XAI is more a clump of scattered works than a structured research program with a solid conceptual and methodological core [99]. In recent years, the need to clarify the concept of opacity has been highlighted by several AI philosophers, who have proposed different definitions and conceptual analyzes aimed at making the term opacity less ambiguous. The first and, arguably, most frequently cited definition is surely that proposed by Paul Humphreys:

> "a process is epistemically opaque relative to a cognitive agent X at time t just in case X does not know all the epistemically relevant elements of the process. A process is essentially epistemically opaque to X if and only if it is impossible, given the nature of X, for X to know all of the epistemically relevant elements of the process" [86, p. 29].

Humphreys' definition highlights two fundamental points:

1. Opacity is an *epistemic* and *agent-relative* property, that is, it concerns the knowledge that some agent possesses of a given computational system;

2. Opacity is *cognitive-dependent*, that is, whether and to what extent a given computational system results opaque to a certain user ultimately depends on the cognitive abilities of the user.

These two points are widely shared by many philosophers participating in the debate on opacity and that take them as starting point of their analyses. For example, Zednick [179] proposes a "pragmatic" and users-oriented account that identifies opacity with the lack of the specific kind of knowledge that the stakeholders of a system require to pursue their aims. A more analytic route is taken by Beisbart [13], who formulates a *Carnapian explication* of epistemic opacity based on the claim that a computational system is opaque to the degree that it is difficult for humans to know and to understand why its outcomes arise. Paez [123] insists on the connection between opacity and pragmatic understanding[4] of AI systems and concludes that the best XAI tools to address opacity are interpretative and approximation methods. Sullivan [153] introduces the notion of *link-uncertainty*, which she defines as a measure of the lack of knowledge necessary to establish to what extend a model is an actual representation of the target-phenomenon it represents. She then argues that scientists in fact consider models generated by a ML system more or less opaque depending on their degree of link-uncertainty with respect to the target-phenomena they represent.

All thee mentioned analyses highlight some characteristics of the opacity problem. In our opinion, however, they do not recognize, at least explicitly, one fundamental aspect of opacity, that is, its plural nature. Looking at the practical contexts in which ML systems are used, we can note that, in fact, there exist various conceptions of opacity, as well as reasons why a ML system is considered opaque by its users in a given context. Identify which are these various conceptions and provide a *taxonomy of opacity* constitutes a crucial philosophical work. Nonetheless, at the moment there are very few available attempts to carry out such a taxonomy. One is that proposed by Burrel [23], who distinguishes between: (i) opacity as intentional corporate or state secrecy, (ii) opacity as technical illiteracy, and (iii) opacity as something that arises from the characteristics of machine learning algorithms and the scale required to apply them usefully'. Another taxonomy has been recently proposed by Creel [34], who distinguishes among *run*, *structural*, and *functional* opacity/transparency. The distinction is based on the intuition that different forms of opacity/transparency occur at different levels of abstraction at which one can understand and describe the structure and functioning of a ML model[5]. Finally, the taxonomy proposed by Boge [21] identifies two forms of opacity: *how-opacity* and *why-opacity*, the former concerning the understanding *of* a ML system and the latter concerning the understanding of a target-phenomenon with (the model generated by) a ML system.

In general, these three taxonomies mark some progress in identifying different forms of opacity. Their analysis, however, remains too broad and needs to be deepened. This section proposes a more detailed analysis that starts from the identification of three macro-

[4]Specifically, he adopts the notion of "pragmatic understanding" originally introduced in [47].

[5]We have already introduced the idea of describing computational systems at different ontological levels of abstraction in Chapter 2, Section 2.1. On the topic, see also [6, 131].

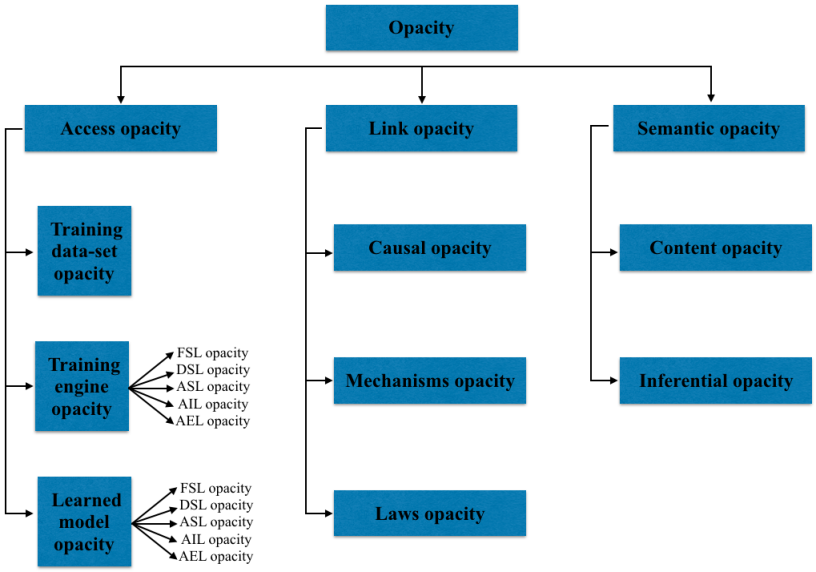


Figure 3.2: Map of the different forms of opacity

dimensions of opacity, called, respectively, *access opacity*, *link opacity* and *semantic opacity*, and then refine further each of them by including specific forms, as depicted in Figure 3.2 (from [62]).

3.3.1 Access Opacity

Access opacity concerns the capability of understanding the structure and functioning of an ML system. It occurs when human users have limited epistemic access to elements that are relevant for explaining, predicting, and controlling the behavior of the considered system.[6] Notice that by "having an epistemic access to an element", we mean the ability to figure out the location of the element and the functional role it plays in the overall structure and functioning of the system.

We identify three main factors that may limit epistemic access and thus cause access opacity.[7] The first coincides with the transparency policies adopted by the system's designers, who might deliberately obscure some relevant details of the system's structure and functioning for either commercial, competition, or privacy reasons. The second is related to the users's background knowledge and skills. Intuitively, the more a user is familiar with a given AI system, the more they can understand, predict and control the system's

[6]The notion of 'epistemically relevant element' is borrowed from [87].
[7]They are related with the three forms of opacity described by Burrel [23].

behavior. Finally, the third arises with the complexity of the system's structure, conceived as a function of both the system's size[8] and format[9] [114]. The intuition is that, as human users possess limited cognitive resources, their ability to explain, predict and control the system's structure and functioning decreases as the complexity of the system increases.

Once clarified these general aspects, we are ready to deepen some details. In the previous section, we explained that ML systems are complex artifacts composed by three different components that ultimately influence their behaviour. For each of these components, we can identify a specific form of access opacity, which manifests itself whenever users' epistemic access to the elements that are relevant to understand the component's structure and functioning, and how it influences the overall behaviour of the system, is limited. In what follows, we provide a more detailed description of each one of these specific forms of access opacity.

Opacity of the Training Sample

This form of access opacity occurs when users have limited epistemic access to relevant structural and semantic properties of the data features included in the samples used to train the system. There are several circumstances where this may happen. A first circumstance is when the system's constructors decide to not adopt data transparency policies, and therefore do not provide (or partially hide) the training sample, in general because of ethical or commercial reasons. A second circumstance is when users have difficulties to interpret the training sample and check the reliability of the data it contains because of its complexity. This scenario is very common when dealing with big data samples.[10] The large size and the variety of data-formats these contain, in fact, makes it hard to check their reliability and identify potential sources of mis-training and biases [115]. A third circumstance occurs when the training process takes place in an open environment, such as the web or the specific part of the world an autonomous robot or a self-driving car is interacting with. In general, to determine in retrospect what data influence the training process in an open environment is practically impossible. The risk that training a system in an open environment produces undetectable biases in the data model that influence the system behavior, therefore, is high. Finally, a fourth circumstance is when stakeholders in charge of surveillance cannot make sense of the data because transformations applied during the construction of the training samples bring them into an incomprehensible format. This scenario is very common when dealing with highly-complex architectures, like deep neural networks (DNN). In several cases, the latter are trained by using low-level *features* that lack any "meaning" for the user and whose reliability is therefore difficult to

[8] I.e., the *number* of elements it includes.

[9] I.e., the *type* of elements it includes and how they are related.

[10] For an overview of the different meanings of the term big data, see: [93].

be analysed and verified [19].

Opacity of the Training Engine

This form of access opacity concerns the capability of users to get epistemic access to elements relevant to understanding the structure and functioning of the training engine and how these influence the overall behaviour of the system. Since the latter can be described at different LoAs, different relevant elements can be identified depending on the LoA considered. At the FSL, the only relevant element is the specification of the optimization method adopted (e.g., supervised vs. unsupervised learning, reinforcement learning etc.). At the DSL, relevant elements include the typology of target function and the heuristic method selected to minimize/maximize it. At the ASL, they include different details of the various routines and sub-routines that allow the machine to practically solve the intended optimization task. These include, for example, procedural techniques to avoid local minima and ensuring that the training terminates only when the global minimum is reached. At the AIL, relevant elements include the specific functions of the program adopted to implement routines and sub-routines, as well as the details concerning the strategies adopted to allocate resources and ensure the feasibility of the implemented procedure. Finally, at the AEL, relevant elements include the various mechanical/electrical components that allow the physical machine to execute the program that implements the procedure solving the intended optimization task.

The various LoAs provide different fine-grained descriptions of the engine's structure and functioning, which may be suitable and relevant for some users and purposes but not for others. For example, consider a data analyst interested in checking a given ML system's training engine to decide whether it can be suitable for a given task of interest (e.g., risk prediction). Likely, the analyst will be interested only in "high-level" details, such as, for example, the kind of loss-function involved and the heuristics adopted to minimize it (DSL), while they will be probably consider irrelevant "low-level" details concerning implementations and executions. Differently, a computer scientist checking the reliability of the training engine will probably be interested in a more fine-grained description that may also include details about the algorithms (ADL), the programs (AIL), and even the hardware (AEL).

The relevance that a certain LoA of the engine has for a user's purposes plays a fundamental role in determining whether and to what extent the training engine will eventually result opaque to that user. In general, we can say that a training engine E is access-opaque (to a certain degree) for a given user u if and only if u has limited epistemic access to the LoAs of E suitable for u's cognitive skills and relevant to u's purposes.

Opacity of the Learned Model

This form of access opacity concerns the capability of users to get epistemic access to elements relevant to understanding the structure and functioning of the optimized computational model resulting from the process of training. As for the opacity of training engine, the opacity of the learned model can occur in different forms, and for different reasons, depending on the LoA considered.

At the FSL, opacity can occur for two maior reasons. First, it can occur because the users have only a partial knowledge of the joint probability distribution $P(X_1, \ldots, X_n, Y)$ that described the intended I/O behaviour of the system, either because this is intentionally hidden or difficult to be reconstructed by the user. Second, it can occur because of the high-complexity of $P(X_1, \ldots, X_n, Y)$, i.e., the high number of variables $X_1, \ldots, X_n$ involved and the difficulty to decompose $P(X_1, \ldots, X_n, Y)$ into simpler joint distributions $P(X_i, Y)$ (with $0 \leq i \leq n$) displaying the dependencies between single features X_i of the input and the system's outcomes. At the DSL, opacity is generally due to the complexity of the model's design, which depends on the incredibly high number of hyper-parameters, parameters, and the non-linear relations among them. On one hand, such complexity makes it difficult for users to get a unitary picture of the model's design, as it happens with simple models, while, on the other hand, it prevents users to figure out what relevant causal role the various elements of the model's design play in the overall I/O behavior of the system.

Finally, different forms of access-opacity of the learned model can be identified at lower LoAs. The latter, however, are not peculiar of ML models and can be identified in all kinds of computational artifacts. For this reason, we skip their discussion here referring the interested reader to existing texts on the topic of opacity in computational models, such as [86, 56, 34, 13].

3.3.2 Link Opacity

Link opacity concerns the use of ML systems to model phenomena in scientific research. It occurs when the model learned by a given ML system conveys inadequate or insufficient information about the elements that are relevant for explaining, predicting, and controlling the target-phenomenon the model aims to represent. We call this form of opacity 'link opacity' to emphasise the fact that it undermines users' ability to establish a link between the model and its target-phenomenon. Stated otherwise, it undermines users' ability to establish whether a model is an 'actual' representation of the target phenomenon or just a possible one. In this regard, the notion of 'link-opacity' resembles that of 'link-uncertainty' introduced by Sullivan, which concerns the extent "to which [a ML] model fails to be empirically supported and adequately linked to the target phenomena" [153,

p. 1].

In general, ML systems are very good at extracting information from large amounts of data and generating highly accurate predictive models without the necessity of background knowledge or human intuition. This ability confers them a clear advantage over more traditional tools in the study of highly complex phenomena (e.g., the fluctuations in financial markets in economics or gene regulation in biology) that represent the target of much contemporary science. For this reason, these systems have quickly spread in several sectors of scientific research, leading to a progressive replacement of the standard scientific methodology[11] with a data-centric approach based on the collection and the AI-supported analysis of observational data [108].

As we have recently argued in [61, 63], we can distinguish between two different kinds of data-centric approaches to scientific research: (i) a data-informed approach, which preserves the classical models and ways of scientific explanation despite the intensive use of ML systems to perform statistical analyses, and (ii) a fully data-driven approach characterized by the replacement of classical "exlanatory" models with the mere predictive ones generated by ML algorithms. While traditionally scientific models represent causal pathways, mechanisms, and laws governing target-phenomena, ML models are nothing more than *phenomenological models*, i.e., representations of the statistical dependencies between observable features of target-phenomena. Hence, although being powerful from a predictive point of view, ML models are often unable to provide sufficient information to explain *why* the target phenomena occur and to figure out ways of controlling them [11]. This point highlights a huge epistemic limitation of the fully data-driven approach. Regardless of whether one takes a realistic or instrumentalist stance towards scientific knowledge, in fact, scientific understanding requires more than mere statistical associations. It needs information about the causes of the phenomena, the mechanisms that produce them, and the laws that regulate their functioning. The lack of this type of information impedes our ability to explain, intervene on, and control the target phenomena, and thus to achieve what De Regt refers to as *pragmatic understanding* [47]. For this reason, when a ML system is unable to provide scientists with information that is essential for the pragmatic understanding of a phenomenon, they tend to consider it as opaque.

Notice that, similar to access opacity, link opacity also occurs in different forms as the elements that are relevant for explaining, predicting, and controlling a given target phenomenon vary depending on the nature of the phenomenon under consideration. In general, we may identify three main forms of link-opacity, each related with one of the three fundamental notions that were previously mentioned: cause, mechanism and law. We refer to these forms, respectively, as *causal opacity*, *opacity of the mechanisms* and *opacity of the laws*.

[11] By *standard scientific methodology* we mean the approach based on the formulation and the experimental evaluation of hypotheses explaining the observable facts.

Causal Opacity

Causal opacity is a direct consequence of the fact that ML systems are unable reconstruct the causal pathways leading to the occurrence of target-phenomena they analyse. As argued in [47], the identification of the causal pathways is essential for the understanding of target-phenomena because it allows scientists:

- to distinguish between the variables necessary and sufficient for the occurrence of the target phenomenon from those that are merely related to it,
- to predict the effects generated by external interventions and, therefore, to understand how to control the target phenomenon by acting on the variables related to it,
- to distinguish between genuine statistical correlations,[12] grounded on the existence of actual cause-effect links, and spurious ones, which are the by-product of statistical paradoxes[13].

The ability of computational systems to recognize causal pathways strictly depends on their ability "to choreograph a parsimonious and modular representation of their environment, interrogate that representation, distort it by acts of imagination and finally answer 'What if?' kind of questions" [126, p.1]. These, in particular, may be either statistical, interventional, or counterfactual questions. The former concern the statistical regularities observed in the naked data and have the form "what if I see x?"; for example: "what if I see salt in the water?". The second one concern the consequences of intervention and has the form "what if I do x?"; for example: "what if I add salt to the water?". Finally, the latter concern some counterfactual state of affairs and have the form "what if I had done x?", for example: "what if I had added salt to the water?", or the contrastive form "what if I had done y instead of x?"; for example: "what if I had added sugar instead of salt?". Causal information is classifiable in terms of the kind of *what-if* questions it can answer. The classification generates a three-layers hierarchy where "questions at the level i (with $i = 1, 2, 3$) can be answered if and only if information from level $j \geq i$ is available" [126, p.1]. The three layers are respectively the *association layer* (AL), the *intervention layer* (IL) and the *counterfactual layer* (CL). Information about statistical regularities is enough for answering questions at the AL and can be inferred directly from the observational data using conditional expectation. At the IL the information requested no longer concerns only what we observe but what we can observe if we perform a certain

[12] We do not mean here that a statistical correlation between a variable x and a phenomenon y is genuine if and only if x is a (proximal or distal) cause of y. Instead, the correlation between x and y is genuine even if x is related to y because of a common cause or effect.

[13] A famous example is the well-known Simpson's paradox, see [127, 128].

action. At the CL, it concerns what we would have observed if a certain condition that did not occur had occurred. We can infer this information by using particular inference engines called *Structural Causal Models* (SCM), which, however, require more than naked data. In particular, they require some background hypotheses encoded in the form of a causal graph and a set of structural equations[14]. Available ML systems usually work at the AL. They do not possess imagination and thus cannot figure out hypotheses beyond the observed data. This inability prevents them from learning causal models and is the reason of their link opacity.

Mechanisms Opacity

In many fields of science, it is common to understand phenomena in terms of mechanisms, i.e., "entities and activities organized in such a way that they are responsible for the phenomenon." [89, p. 120]. The reason is that thinking in terms of mechanisms presents some clear epistemological advantages. It permits to manage with complexity and lead highly-complex phenomena back to simpler, more fundamental facts [12]. It allows us to provide an explanation by stating a description, as "[by] providing a description of the mechanism responsible for a phenomenon, one provides an explanation for *why* that particular phenomenon occurs and *why* it has the proprieties it does" [77, p.217]. Finally, it supports generalization because mechanisms "work in the same or similar way under the same or similar conditions" [33, p.19]. Formulating a mechanistic explanation, however, needs much more than mere observational data. It requires to hypothesize what simpler, more fundamental entities and activities may produce the target phenomenon by interacting with one another. The reason is that mechanistic thinking relies on heuristics that are very different from those used to train ML systems. Actually, the nature of these heuristics is a matter of debate. In [12], Bechtel and Richardson identify two mains reasoning strategies followed by scientists to identify mechanisms' structure and functioning, which they name *decomposition* and *localization*. Roughly, the former consists of decomposing the overall phenomenon into low-level activities while the latter consists of localizing these activities in components of the system identified as responsible for producing the target phenomenon. A different account of mechanistic reasoning is proposed in [33]. According to these authors, mechanistic reasoning is a hypothesis-driven practice that combines scientific exploration, hypotheses-formulation and experimental manipulation. Loosely speaking, the search for mechanisms is an iterative process consisting of the iterated application of specific reasoning and experimental techniques that allow scientists to refine a raw hypothesis about the mechanism's structure and functioning, generally in the form of a sketch representation full of black boxes, until obtaining a sufficiently clear and detailed description. Regardless of the details, in both cases, the information neces-

[14]For more details on this topic we refer to [125, 127, 128, 126].

sary to understand a mechanism requires hypotheses that cannot be inferred from mere observational data but need a fundamental contribution of imagination. Since the heuristics implemented in AI systems are unable to formulate this type of hypotheses, these systems cannot generate mechanistic explanations, and are eventually to be considered link-opaque.

Laws Opacity

Since its birth, discovering the laws that govern phenomena has represented a fundamental aim of science. From an epistemological point of view, scientific laws are essential to the understanding of phenomena. They allow scientists to explain *why* phenomena occur in a way rather than another, to predict under what circumstances they occurs, and to figure out how to act for controlling their occurrence. Philosophers of science have long debated the nature of laws, taking sides on two opposing positions: the instrumentalist and the realist.[15] A detailed discussion of these specific positions is beyond the scope of this work. Here we simply note that, in scientific practice, the term 'scientific law' may refer to different things. In some cases, 'law' denote sentences that describe mere patterns of regularities between observable variables. An example is Charles' law in thermodynamics, which shows the relationship between the volume and the temperature of a gas. These kinds of laws do not substantially differ from the functions learned by ML algorithms and, indeed, a ML system might easily infer Charles' law by analyzing a sufficiently large sample of data. In other cases, a law is instead a description of the structural relationships between observable variables and variables that:

- denote unobservable entities, whose existence scientists theoretically hypothesize but cannot statistically infer from observational data,
- scientists consider the main causes of the target class of phenomena.

Gauss's law, which relates the electric charge and the magnitude of the electric field[16], is an example of the latter.

Both types of law coexist in scientific practice, but scientists tend to consider laws of the second type epistemologically more relevant. Interestingly, the reason is not that they believe in the actual existence of unobservable entities, but because these laws allows them to bring the observed phenomena back into a single representation of reality and figure out how to control their occurrence. The epistemological value of these laws is therefore independent from the 'realists vs instrumentalists' debate and have pragmatical roots.

[15] On the debate about instrumentalists and realists, see [132].

[16] The electric field is an unobservable entity theoretically hypothesized to explain remote interaction among particles.

For this reason, a science including only the first type of laws is very difficult, and maybe impossible, to imagine.

Unfortunately for ML systems, the identification of laws of the second kind is a purely theoretical work. It relies on the human mind's ability to go beyond the observable phenomena and figure out in what the supposed basic structure of reality might consist. ML systems do not possess this ability, and as a result scientists may regard them as opaque.

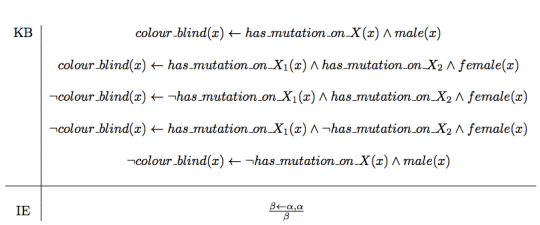
3.3.3 Semantic Opacity

In information theory, it is common to distinguish between a *structural* and *semantic* aspect of information. The former concerns the mathematical and physical properties of information, whereas the latter concerns its meaning. In the case of ML systems, the structural aspect coincides with the properties of the model that the system learns from data, which can be specified at different LoAs as explained in Section 3.3.1. These properties are relevant for understanding how the learned model works and, therefore, are connected with the problem of access opacity mentioned above. Stated otherwise, the opacity of the structural aspects of information is a form of access-opacity and, more specifically, an occurrence of access-opacity of the learned model.

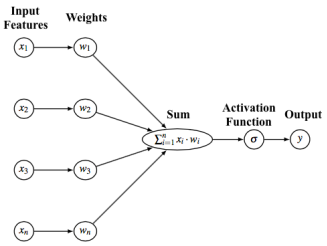
Differently from structural aspects, semantic aspects coincide with the potential semantic contents of the information stored by the learned model. Although the latter may be not directly relevant for determining the functioning of the model, their understanding is fundamental for users to grasp and interpret the information that the system learns and manipulates. In fact, it happens that if the format used to store and manipulate information prevents users from giving it a meaningful interpretation, then users deem the system opaque. This sense of "opacity", however, cannot be included into any of the kinds described so far. It represents a new form of opacity that we call *semantic opacity*.

Semantic opacity can occur in three different circumstances. First, when the learned model lacks a clear, well-defined semantic interpretation that allows users to make sense of both the information it stores and the inferences it performs. Second, it may take place when a semantic for the learned model is available but it is not comprehensible because of the users' limited cognitive resources, inadequate background knowledge, or lack of relevant epistemic skills. Third, it can arise when the semantics of the learned model provides the stored information with a meaning that is inadequate for the context.

In what follows we distinguish between two forms of semantic opacity. The first one concerns the content of the information learned by a ML system, whereas the second one concerns the inferences it uses to manipulate such information. We refer to these two distinct forms of semantic opacity as, respectively, *content opacity* and *inferential opacity* respectively.



(a) System A



(b) System B

Figure 3.3: Two expert systems

Content Opacity

This form of opacity occurs when the format used by an ML system to store information prevents users from grasping its semantic content and using it for their purposes. Notice that each type of ML system adopts a peculiar format to represent the information learned from the data. In general, the choice of the format affects the users' ability to provide the stored information with an interpretation that allows them to grasp its semantic content.

By way of example, let us compare the two expert systems A and B reported in the Figures 3.3a and 3.3b.

A is a (very trivial) example of rules-based system that can be generated via inductive logic-programming and used in the context of medical decision-making to predict whether a patient will suffer from colour blindness. In ML models such as A, the information learned from data is stored by means of logical sentences stemming from a given formal language that are called *hypotheses*, e.g.:

$$colour\_blind(x) \leftarrow has\_mutation\_on\_X(x) \wedge male(x)$$

Hypotheses are collected in the knowledge base (KB) and manipulated through iterative applications of the rules included in the inference engine (IE) in order to generate predictive outcomes. It is easy to see how a standard Tarskian semantics, which maps the syntactic elements (i.e., predicates, variables, quantifiers, Boolean connectives) to features relevant to the context (i.e., genetic mutations, sex, disease, patients), may easily provide the information stored in the KB with a meaningful interpretation that allow users to grasp its semantic content.

Things are different with a ML model such as B, which stores the information learned from data by means of the "weights", i.e., the numerical parameters connecting the various nodes in the network. Providing these parameters with a clear semantic interpretation is difficult as they usually have a mere statistical instrumental meaning. That is, their values have nothing to do with the phenomenon the model represents, they are simply

those values that allow the network to minimize the prediction error [11].

Inferential Opacity

This form of opacity occurs when the format of the inferences used by a ML system to manipulate information prevents users from making sense of the reasoning paths it follows. As for the format used to represent the information learned from the data, each type of ML system uses a specific kind of inferences to manipulate information. For instance, a rules-based system such as A in the example above manipulates information by applying the rules included in the IE to the hypotheses stored in the KB. Conversely, a DNN such as B manipulates the information using analytical calculations that merge input data and learned parameters. Unfortunately, it is not always possible to provide inferences with an interpretation meaningful for the context of use that allows users to reconstruct the reasoning pathways followed by the system in humanly understandable terms. In some cases, inferences have a purely instrumental value, i.e., they allow the system to generate accurate predictive outcomes, but lack of any meaningful semantic interpretation. In other cases, they may posses a well-defined and meaningful semantics that, however, is incomprehensible to a stakeholder because of their limited cognitive resources, their inadequate background knowledge or their lack of fundamental skills. In all these circumstances, we can say that the inferences performed by the system under consideration make the latter "semantically" opaque.

3.4 Explainable Artificial Intelligence

Explainable Artificial Intelligence is a recently born and thriving area of research that is generating increasing interest both within and outside the AI community. Despite this, it does not appear as a structured research field, with a well-defined methodology, but more as a miscellany of different methods and techniques, each one with its own strategy to counter opacity and make ML systems' behaviours and outcomes more "explainable" to users [1, 76, 8, 121]. For the purposes of this work, we restrict our focus only to a specific class of XAI methods, so-called *post-hoc explanation methods* (PEMs). In very general terms, a PEM is a procedure that takes an existing target (opaque) ML system and generates a *post-hoc* explanation of either its behaviour or some of its outcomes. XAI literature recognizes the existence of two main classes of PEMs [121]:

Features Selection Methods (FSMs). This class of methods includes algorithms and procedures to identify the features in the input space of the target system that are chiefly responsible for a given outcome. In general, these methods are considered particularly

useful to reduce the complexity of the system's input space and make easier for humans to identify the data responsible for selected outcomes. There exists a wide variety of FSMs adopted in XAI (for a survey, see [76]). The simplest FSMs consist of assigning a numerical score to features that highlight their relevance for a given prediction. Saliency masks and heath-maps consist of visual representations highlighting the areas of the input space that mainly contribute to a given prediction. For this reason, they are particularly suitable for tasks related to image classifications. Sensitivity analysis seeks to understand the relevance of specific features by locally perturbing the input space around a feature and observing to what extent it affects the predictive outcomes. Partial dependence and accumulated local effects plots show the marginal effect that singular features have on a specific outcome of the ML system [72]. In general, they are particularly useful for understanding whether the dependency between the feature and the output is linear, monotonic or more complex. Functional decomposition, finally, includes a class of methods to deconstruct the usually high-dimensional functions learned by machine learning algorithms and express them as a sum of weighted individual features that can be easily visualised by humans [121]

Surrogate Models (SMs). These methods rely on learning *post-hoc* transparent models that emulate the I/O behaviour of the target system. Explanations provided by surrogate models can be of two kinds:

- *Global explanations* concern the overall strucuture and functioning of a given target-system. They typically consist of surrogate models that emulate globally the properties and behaviour of the target-system and are easily understandable by users.

- *Local explanations* concern a single outcome of the target-system. They typically consist of surrogate models that emulate the properties and behaviour of the target-system locally on a defined neighborhood of the concerned I/O pair.

Typically, surrogate models are trained with supervised learning methods applied to a synthetic training sample including examples of the I/O behaviours of the target system. That is, given a target-system M whose I/O behaviour is described by the function $f_W^M : \mathcal{X} \to \mathcal{Y}$, a synthetic training sample is a finite collections of pairs $\langle X, f_W^M(X)\rangle$, $X \in \mathcal{X}$. For global explanations, this collection includes examples for each input $X \in \mathcal{X}$; for local explanations, it includes only a local neighbourhood of the outcome of interest. Different ML architectures have been used as surrogate models in XAI literature: linear models [140], decision trees [124], formal arguments [169], logic programs learned through inductive logic programming [37], logical circuits [36], and probabilistic circuits [176].

3.4.1 Reliability Properties of *post-hoc* Explanations

In the following sections, we focus specifically on post-hoc explanation methods based on surrogate models. Our purpose, in particular, is to define a method for verifying that post-hoc explanations conveyed by surrogate models are effective and reliable. To this aim, we apply methods and techniques from probabilistic model checking introduced in the previous chapters. Our starting point consists in identifying (and "informally" defining) a set of desirable properties that a surrogate model should satisfy in order to provide effective and reliable explanations. We refer to the latter collectively as *explanation reliability properties*.

In general, three explanation reliability properties are usually mentioned in XAI literature [110, 75, 5, 143, 70, 91]:

- **Transparency**. This property is usually defined as a measure of "to what extent the properties and behaviours of a computational system are accessible to its human users" [76, p.6]. In this regard, transparency can be seen the "dual" of access opacity we analysed in Section 3.3. Like the latter, transparency is a *relational* property that depends on both the structure of a computational system and the epistemic abilities of its users. It is also a "fuzzy" property insofar systems are rarely fully opaque/transparent to a users, while they are typically transparent *to some degree*, which depends on how much the users is able to get access to the properties and behaviours of the system itself. In this regard, notice that it is not reasonable to require a surrogate model to be completely transparent to provide effective explanations. What is required, more typically, is that the surrogate model is more transparent than the target ML system it aims to explain. Intuitively, we can say, the more a surrogate model is transparent to a given user with respect to the target system it aims to explain, the more effective will be its explanations. This intuition will be exploited in the next sections to provide a more formal definition of transparency for surrogate models.

- **Accuracy**. This property is usually defined as a measure of to what extent a surrogate model is able to accurately imitate the properties and behaviour of the opaque target-system [76, p. 7]. In XAI literature, there exist a variety of methods to measure the accuracy of surrogate models, like the *accuracy-score* or the *F1-score* (see, [110, 75]). All these methods are eventually based on *output fidelity*, i.e., on a measure of the differences between the observable outcomes of the surrogate model and those of the target-system on a fixed observable in a given experimental setting. Output fidelity is thus an experimental measure, which is obtained by iteratively running both the target-system and the surrogate model and thus confronting their outcomes. In this regard, output fidelity is limited both by the fact that it focuses

on just I/O behaviours and that its reliability strictly depends on the number of runs executed in the experimental setting. Here we propose a different strategy to measure accuracy based on a state-by-state *full exploration* of both the target-system and the surrogate model and on a subsequent comparison of their properties, as better detailed in Section 3.5.2.

- **Trustworthiness**. Differently from accuracy, it not very clear from AI literature what should mean for a model to be *trustworthy*. The debate on trustworthy AI is wide and full of many different definitions for this property. Some of them involve social and legal dimensions (e.g., [4]), hence locating trustworthiness at least partly outside the realm of formal properties that can be checked automatically. Others (e.g., [65]) strictly relates the notion of trustworthiness to that of *computational reliability* [56, 57], according to which a computational process is *reliable* if and only if *it produces correct outputs most of the time*. Following this second line of thought, we introduce here a "minimal"[17] notion of trustworhtiness specific for XAI-explanations of ML systems' behaviours based on surrogate models. According to this definition, the explanation of a behaviour b (of an opaque ML target-system to be explained) provided via a surrogate model is *trustworthy* if and only if, given that the target-system exhibits the behaviour b, then the surrogate model is likely to exhibit the same behaviour. This quite intuitive characterization is thus made more rigorous via formally defining trustworthiness as the *probability that a surrogate model exhibits behaviour b given that the target-system exhibits b*. In what follows, we will show how this probability can be further characterized and computed via probabilistic model checking instruments and methods.

The three characterizations of transparency, accuracy, and trustworthiness above introduced are informal. Our next step is to introduce a proper mathematical framework in which to define them more rigorously. First of all, we note that all three properties are *relational*, i.e., they are not satisfied by a system but by a relation between two systems, or between a system and one or more of its users. In this regard, canonical Markovian frameworks introduced in Chapter 1 and commonly used in probabilistic model checking are not very suitable, as they are mostly conceived to manage "intrinsic" rather than relation properties. More suitable are instead *multi-agent* frameworks (see, Sections 2.11 and 2.10) that allow to represent and analyse specific relations holding between different kinds of agents. In our framework, we have to account for three different kinds of agents:

(i) the opaque ML target-system to be explained (henceforth, target-system),

[17] i.e., a notion that can be enriched and extended in future works to account for the on-going developments in the debate.

(ii) the surrogate models explaining it, and

(iii) the users interacting with both the target-system and the surrogate models and to whom explanations are directed.

We do not consider all the components of these agents, but only those which are relevant to determine properties of transparency, accuracy and trustworthiness. Both the target-system and the surrogate models are kinds of ML systems, their general architecture therefore includes three main components (see, Section 3.2): (i) the training sample, (ii) the training engine, and (iii) the learned model. Here, we focus only on the learned model, which is responsible for the I/O behaviour of the trained system. This choice is motivated by the fact that properties of accuracy and trustworthiness are typically measured on already trained target-systems and surrogate models; details on the training sample and the process of training are therefore irrelevant and can be omitted. Furthermore, we focus specifically on high-level properties of systems, notably related to the functional and design specification levels, while we do not consider properties characteristic of lower levels of abstraction. On the one hand, this is a methodological choice related to intrinsic limitations of the model checking techniques we adopt. As explained more in details in Chapter 2, standard probabilistic model checking relies on formal-mathematical models of computational systems that focus on design specifications and abstract away from details concerning implementations and executions. This makes model checking more adapt to verify functional and design properties. On the other hand, we choose to focus specifically on functional and design specification levels because these are the most relevant for the majority of XAI applications. In this regard, notice that similarities between target-systems and their surrogate models typically concern only the functional specification level (i.e., the I/O mappings), while they already differ quite a lot at the level of design. At the same time, contrary to other XAI techniques like *inspection methods* (see, [1]), which are conceived to get insights about the low-level procedures governing opaque ML systems, post-hoc explanation methods are mostly conceived to work at the FSL, enabling users to visualize and understand the information included in the (highly-complex) joint distribution $P(X_1, \ldots, X_n, Y)$.

In our framework, both the target-system and the surrogate models are modelled by means of *discrete-time Markov chains*. This is a quite standard choice motivated by the fact that both are kinds of stochastic computational systems, which are commonly represented in probabilistic model checking via finite Markov models like discrete-time Markov chains. Concerning users, what is relevant of them is their ability "to get epistemic access" to relevant behaviour of the systems. We can model this ability through epistemic accessibility relations usually involved in epistemic logic. In our case, such accessibility relations will be defined over the possible states of the target-system, respectively, the surrogate models of interest.

Based on this framework, we introduce specific formalizations of the explanatory reliability properties above mentioned. As in [158], properties of accuracy and trustworthiness are here defined as two different measures of the "distance" between the formal properties that the target-system and its surrogate models satisfy locally on a given state. Transparency is defined as a normalized measure of the number of states that satisfy a certain property of interest epistemically accessible for a given user (or group of users) with respect to the total number of states in the system that satisfy that property. We then include a fourth property of interest that summarizes all the others and that we call *explanatory power*. The latter will be defined as a normalized measure of how much transparency, accuracy and trustworthiness of a given surrogate model contribute to make the explanation it provides suitable for a given user.

To formulate degrees of transparency, accuracy, trustworthiness, and explanatory power, we introduce a new language that extends standard PCTL with appropriate operators. Furthermore, we develop specific algorithms to check whether models formalized in our framework satisfy explanation reliability properties specified by means of the new logic we introduce. These algorithms will be obtained as extensions of the standard model-checking procedures for PCTL introduced in Chapter 2.

3.5 A Multi-Agent Semantics for Explanation Reliability Properties

Let us consider a multi-agent structure including a finite non-empty set $\mathcal{A}$ of *models* and a finite non-empty set $\mathbf{U}$ of *users*. We denote generic elements of the set of models $\mathcal{A}$ by lower case letters $i, j, \ldots$. These can be either *learned models* generated by a given target-system (henceforth, *target-models*), or *surrogate-models*, i.e., models that are used to post-hoc explain the behaviour of the target-system to the users. Notice that, the notation $i, j \in \mathcal{A}$ does not explicitly distinguish among target and surrogate models. The reason is that whether a certain model is a target or a surrogate model depends on the specific context of application. For example, we might use a Bayesian network either as a surrogate model to explain the behaviour of a deep neural network or as a target model whose behaviour can be explained using a probabilistic rules-based system. The models $i \in \mathcal{A}$ are formally specified via labelled DTMCs, i.e., $i := \langle \mathcal{S}, T^i, AP^i, l^i \rangle$ for each $i \in \mathcal{A}$. For the sake of simplicity, we assume that the state space $\mathcal{S}$ is the same for all the $i \in \mathcal{A}$, while the transition matrix T^i, the set of labels AP^i and the labelling l^i are local to each $i \in \mathcal{A}$. On the other hand, we denote by u a generic element of the set of users $\mathbf{U}$. The abilities of users $u \in \mathbf{U}$ to get *epistemic access* to the various alternative states of the models are specified in terms of *epistemic accessibility relations* $\sim^u$ between pairs of states $s, s' \in \mathcal{S} \times \mathcal{S}$. We can read informally the notation $s \sim^u s'$ as saying "the state s' is

epistemically accessible from s for agent u". Furthermore, we associate to each $u \in \mathbf{U}$ a vector of weights $[x, y, z]^u$ whose elements x, y and z are values in the interval $[0, 1]$ such that $x + y + z = 1$. These weights are used to model the *explanatory relevance* that user u attributes to, respectively, transparency, accuracy and trustworhtiness of surrogate models. Notice that by explanatory relevance we mean a measure of how much relevant the user u considers each one of those three properties with respect to the others for the task of explaining the behaviour of opaque models. By convention, we assume that the first element $x \in [x, y, z]$ denotes the explanatory relevance of transparency, the second element $y \in [x, y, z]$ that of accuracy, and the third element $z \in [x, y, z]$ that of trustworthiness. For example, if we associate the user u with the vector $[0.2, 0.9, 0.9]^u$, we specify that, according to u, accuracy and trustworhtiness are really more relevant for explaining the behaviour of a target-model than transparency (0.9 vs. 0.2).

3.5.1 Ex-PCTL Syntax

We now introduce the syntax of *Explanatory Probabilistic Computation Tree Logic* (Ex-PCTL), which can be used to specify transparency, accuracy, trustworhtiness and explanatory power of surrogate models against behaviours and properties of target models for specific single or groups of users. Models' properties and behaviours are specified in this logic via standard PCTL formulae.

Definition 52 (Ex-PCTL Syntax)

$$\mathcal{A} := \{i, j, \dots\}$$
$$\mathbf{U} := \{u_1, u_2, \dots\}$$
$$\nabla := <, \leq, =, \geq, >$$
$$\phi := \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid P_{\nabla b}\psi$$
$$\psi := \bigcirc\phi \mid \phi_1 \bigcup \phi_2 \mid \phi_1 \overset{\leq t}{\bigcup} \phi_2$$
$$\theta := V^{i,u}_{\nabla h}\phi \mid CV^{i,u}_{\nabla h}\phi \mid DV^{i,u}_{\nabla h}\phi$$
$$\alpha := A^{i,j}_{\nabla h}\phi \mid T^{i,j}_{\nabla h}\phi \mid$$
$$\epsilon := Ex^{i,j,u}_{\nabla h}\phi \mid CEx^{i,j,\Gamma}_{\nabla h}\phi \mid DEx^{i,j,\Gamma}_{\nabla h}\phi$$

Superscript indices of formulae denote either models $i \in \mathcal{A}$, single users $u \in \mathbf{U}$, or groups of users $\Gamma \subseteq \mathbf{U}$. The notation ∇ is a meta-variable for $<$, $\leq$, $=$, $\geq$ or $>$. Parameters b and h denote real numbers ranging in the interval $[0, 1]$ and modeling probabilities. Formulae ϕ and ψ include usual PCTL formulae with their standard reading (see, Section 2.7.1) and specifying characteristic properties and behaviours of models $i \in \mathcal{A}$.

The θ-formulae include different operators to specify degrees of transparency of models for a single user $u \in \mathbf{U}$ or a group of users $\Gamma \subseteq \mathbf{U}$. Their informal readings are as follows:

- $V^{i,u}_{\nabla h}\phi$: the degree of transparency of model i against property ϕ according to user u is less/equal/greater than h;
- $CV^{i,\Gamma}_{\nabla h}\phi$: the common degree of transparency of model i against property ϕ according to the group of users $\Gamma \subseteq \mathbf{U}$ is less/equal/greater than h;
- $DV^{i,\Gamma}_{\nabla h}\phi$: the distributed degree of transparency of model i against property ϕ according to the group of users $\Gamma \subseteq \mathbf{U}$ is less/equal/greater than h.

The α-formulae include different operators to specify degree of accuracy and trustworhtiness of a surrogate model $i \in \mathcal{A}$ explaining a property ϕ of an opaque target-model $j \in \mathcal{A}$. Their informal reading is as follows:

- $A^{i,j}_{\nabla h}\phi$: Surrogate model i explains property ϕ of model j with a degree of accuracy less/equal/greater than h;
- $T^{i,j}_{\nabla h}\phi$: Surrogate model i explains property ϕ of model j with a degree of trustworthiness less/equal/greater than h.

Finally, the-ϵ formulae include different operators to represent degrees of explanatory power of a surrogate model $i \in \mathcal{A}$ against property ϕ of a target-model $j \in \mathcal{A}$ computed either for a single user $u \in \mathbf{U}$ or for a group of users $\Gamma \subseteq \mathbf{U}$. Their informal reading is as follows:

- $Ex^{i,j,u}_{\nabla h}\phi$: The degree of explanatory power of surrogate model i against property ϕ of target-model j according to user u is less/equal/greater than h;
- $CEx^{i,j,\Gamma}_{\nabla h}\phi$: The common degree of explanatory power of surrogate model i against property ϕ of target-model j according to the group of users Γ is less/equal/greater than h;
- $DEx^{i,j,\Gamma}_{\nabla h}\phi$: The distributed degree of explanatory power of surrogate model i against property ϕ of target-model j according to the group of users Γ is less/equal/greater than h.

3.5.2 Ex-PCTL Semantics

The satisfiability conditions for Ex-PCTL formulae are defined on a multi-agent structure that we call *target-surrogate-users systems* (TSU, for short) and defined as follows:

Definition 53 (Target-surrogate-users system) *A TUS is a tuple:*

$$\mathcal{M}_{TUS} := \langle \mathcal{S}, \mathcal{A}, \mathbf{U}, \{T^i\}_{i\in\mathcal{A}}, \bigcup_{i\in\mathcal{A}} AP^i, \{l^i\}_{i\in\mathcal{A}}, \{\sim^u\}_{u\in\mathbf{U}}, \{[x,y,z]^u\}_{u\in\mathbf{U}}\rangle$$

including a finite non-empty set of states $\mathcal{S}$, a finite non-empty set of models $\mathcal{A}$, a finite non-empty set of users $\mathbf{U}$, a family $\{T^i\}_{i\in\mathcal{A}}$ of transition matrices $T^i : \mathcal{S}\times\mathcal{S}\mapsto[0,1]$, one for each model $i\in\mathcal{A}$, a set of labels $\bigcup_{i\in\mathcal{A}} AP^i$ obtained as the union set of all the sets of labels AP^i of each model $i\in\mathcal{A}$, a family $\{l^i\}_{i\in\mathcal{A}}$ of labelling functions $l^i : \mathcal{S}\mapsto 2^{AP^i}$, one for each model $i\in\mathcal{A}$, a family $\{\sim^u\}_{u\in\mathbf{U}}$ of epistemic accessibility relations, one for each user $u\in\mathbf{U}$, and a family $\{[x,y,z]^u\}_{u\in\mathbf{U}}$ of vectors of weights to model explanatory relevance of properties, one for each user $u\in\mathbf{U}$.

3.5.3 Satisfiability of ϕ and ψ-formulae

ϕ-formulae specify *states-properties* of either target or surrogate models $i\in\mathcal{A}$. Their satisfiability conditions are hence defined with respect to a model $i\in\mathcal{A}$ and a state $s\in\mathcal{S}$. Instead, ψ-formulae specify *paths-properties* of models, i.e., properties concerning possible evolutions across time of a model $i\in\mathcal{A}$. Their satisfiability conditions are defined with respect to a model $i\in\mathcal{A}$ and a path $\pi\in\Pi$. In general, satisfiability conditions for ϕ- and ψ-formulae are analogous to those of standard PCTL reported in Section 2.7.2.

Definition 54 (Satisfiability of ϕ and ψ formulae) *Given a model $i\in\mathcal{A}$ and a state $s\in\mathcal{S}$, respectively, a path $\pi\in\Pi$, the following conditions hold:*

$$
\begin{aligned}
&i,s \models \top, \quad \forall s\in\mathcal{S}\\
&i,s \models p \ \text{ iff } \ p\in l^i(s)\\
&i,s \models \phi_1\wedge\phi_2 \ \text{ iff } \ i,s\models_i \phi_1 \text{ and } i,s\models\phi_2\\
&i,s \models \neg\phi \text{ iff } i,s\not\models\phi\\
&i,\pi \models \bigcirc\phi \text{ iff } i,\pi(1)\models\phi\\
&i,\pi \models \phi_1 \bigcup^{\leq t} \phi_2 \text{ iff } \exists\tau\leq t : i,\pi(\tau)\models\phi_2 \text{ and } \forall\tau' : 0\leq\tau'<\tau,\ i,\pi(\tau')\models\phi_1\\
&i,\pi \models \phi_1 \bigcup \phi_2 \text{ iff } \exists\tau\geq 0 : i,\pi(\tau)\models\phi_2 \text{ and } \forall\tau' : 0\leq\tau'<\tau,\ i,\pi(\tau')\models\phi_1\\
&i,s \models P_{\nabla b}\psi \text{ iff } P(i,s\models\psi)\nabla b\,,
\end{aligned}
$$

where $P(i,s\models\psi)$ denotes the probability that a path π originating in s (i.e., such that $\pi(0)=s$) satisfies ψ according to the transition matrix T^i. The methods to compute this probability are the same used in standard PCTL to compute $P(s\models\psi)$, see Section 2.7.3.

3.5.4 Satisfiability of θ-formulae

We interpret the degree of transparency of a model $i \in \mathcal{A}$ against property ϕ for an user $u \in \mathbf{U}$ in function of an initial state $s \in \mathcal{S}$, here denoted by $V_\phi^{i,u}(s)$, as a measure of the ability of u to gain epistemic access from $s \in \mathcal{S}$ to the various states of i satisfying ϕ:

$$V_\phi^u(s) := \begin{cases} \frac{|Sat_i(\phi) \cap eq^{\sim^u}(s)\}|}{|Sat_i(\phi)|} & \text{if } \; Sat_i(\phi) \neq \emptyset\,, \\ 1 & \text{otherwise}\,, \end{cases} \tag{3.1}$$

where $Sat_i(\phi) := \{s \in \mathcal{S} : i, s \models \phi\}$ and $eq^{\sim^u}(s) := \{s' \in \mathcal{S} : s \sim^u s'\}$.

In other words, the degree of transparency is computed as the ratio between the cardinality of the set of states of i that satisfy ϕ and are epistemically accessible from $s \in \mathcal{S}$ and the cardinality of the whole set of states of i that satisfy ϕ. Clearly, when all the states that satisfy ϕ are accessible, we have that $V_\phi^{i,u}(s) = 1$ and the model i is *fully transparent*. In all the other cases, the higher the number of states satisfying ϕ that are epistemically accessible, the higher the degree of transparency. A particular scenario is when $Sat_i(\phi) = \emptyset$. In this case, the model never satisfies the property of interest and we assume, by convention, $V_\phi^u(s) = 1$. An appropriate satisfiability condition can now be introduced as follows.

Definition 55 (Satisfiability of transparency formulae) *Given a TUS $\mathcal{M}_{TUS}$, a model $i \in \mathcal{A}$, a user $u \in \mathbf{U}$ and a state $s \in \mathcal{S}$ the following condition holds:*

$$\mathcal{M}_{TUS}, s \models V_{\nabla h}^{i,u}\phi \;\text{ iff }\; V_\phi^{i,u}(s)\ \nabla h \tag{3.2}$$

For modeling *common* and *distributed* degrees of transparency, we introduce specific epistemic accessibility relations for groups of users $\Gamma \subseteq \mathbf{U}$, defined as follows:

- $\sim_C^\Gamma := it(\bigcup_{u\in\Gamma} \sim^u)$ where *it* denotes the *iterative closure*;
- $\sim_D^\Gamma := \bigcap_{u\in\Gamma} \sim^u$.

These definitions are analogous to common and distributed knowledge clauses used in standard CTLK (see, Section 2.9). C ommon $CV_\phi^{\Gamma,u}(s)$ and distributed $DV_\phi^{\Gamma,u}(s)$ degrees of transparency for groups of users $\Gamma \subseteq \mathbf{U}$ are accordingly defined as follows:

$$CV_\phi^{i,\Gamma}(s) := \begin{cases} \frac{|Sat_i(\phi) \cap eq^{\sim_C^\Gamma}(s)\}|}{|Sat_i(\phi)|} & \text{if } \; Sat_i(\phi) \neq \emptyset\,, \\ 1 & \text{otherwise}\,, \end{cases} \tag{3.3}$$

$$DV_\phi^{i,\Gamma}(s) := \begin{cases} \frac{|Sat_i(\phi) \cap eq^{\sim_D^\Gamma}(s)\}|}{|Sat_i(\phi)|} & \text{if } \; Sat_i(\phi) \neq \emptyset\,, \\ 1 & \text{otherwise}\,. \end{cases} \tag{3.4}$$

To conclude, let us define the satisfiability conditions for θ-formulae as follows.

Definition 56 (Satisfiability of θ formulae) *Given a TUS $\mathcal{M}_{TUS}$, a model $i \in \mathcal{A}$, a group of users $\Gamma \subseteq \mathbf{U}$ and a state $s \in \mathcal{S}$, the following conditions hold:*

$$\mathcal{M}_{TUS}, s \models CV^{i,\Gamma}_{\nabla h}\phi \;\text{ iff }\; CV^{\Gamma,i}_{\phi}(s)\ \nabla h\,, \tag{3.5}$$

$$\mathcal{M}_{TUS}, s \models DV^{i,\Gamma}_{\nabla h}\phi \;\text{ iff }\; DV^{\Gamma,i}_{\phi}(s)\ \nabla h\,. \tag{3.6}$$

3.5.5 Satisfiability of α formulae

Accuracy

We interpret the degree of accuracy as a *Jaccard index*. Given two sets A and B, the *Jaccard index* of A and B, denoted by $J(A, B)$, is defined as:

$$J(A, B) := \frac{\mid A \cap B \mid}{\mid A \cup B \mid}\,, \tag{3.7}$$

when $A \cup B \neq \emptyset$, and $J(A, B) := 1$ otherwise. Given a computational model $i \in \mathcal{A}$ and an Ex-PCTL formula ϕ, we define $Sat_i(\phi)$ as the set of states $s \in \mathcal{S}$ such that $i, s \models \phi$. Given a state $s \in \mathcal{S}$, we denote by $Reach^i(s)$ the set of all the states $s' \in \mathcal{S}$ that are almost surely reachable from s for the agent $i \in \mathcal{A}$. A state $s' \in \mathcal{S}$ is *almost surely reachable* from another state $s \in \mathcal{S}$ according to an agent $i \in \mathcal{A}$ if and only if

$$(h)^i_{\{s'\}}(s) = 1\,, \tag{3.8}$$

where $(h)^i_{\{s'\}}(s)$ is the hitting probability of the event $\{s'\}$ computed through the transition matrix T^i as by Equation (1.8).

Given a property ϕ and an initial state $s \in \mathcal{S}$, we denote by $\Phi_i(s)$ the set of states that satisfy ϕ and are almost surely reachable from s by i, i.e.:

$$\Phi_i(s) := Sat_i(\phi) \cap Reach^i(s)\,. \tag{3.9}$$

Given two models $i, j \in \mathcal{A}$ such that i is a surrogate model and j is the opaque target model to be explained, we define the *degree of accuracy* of surrogate model i in emulating property ϕ of opaque target model j with respect to a given initial state $s \in \mathcal{S}$, denoted by $A^{i,j}_{\phi}(s)$ as follows:

$$A^{i,j}_{\phi}(s) := J(\Phi_i(s), \Phi_j(s))\,. \tag{3.10}$$

The satisfiability conditions of Accuracy formulae is defined as follows:

Definition 57 (Satisfiability of accuracy formulae) *Given a TUS $\mathcal{M}_{TUS}$, a surrogate model $i \in \mathcal{A}$, a target model $j \in \mathcal{A}$, and a state $s \in \mathcal{S}$, the following condition holds:*

$$\mathcal{M}_{TUS}, s \models A^{i,j}_{\nabla h}\phi \;\text{ iff }\; A^{i,j}_{\phi}(s)\ \nabla h\,. \tag{3.11}$$

Trustworthiness

We define trustworthiness as a probability. In particular, the degree of trustworthiness of a surrogate model $i \in \mathcal{A}$ in emulating a property ϕ of a model j, denoted by $T^{i,j}_{\phi}(s)$, is defined as the probability that eventually j will reach a state satisfying ϕ, provided that the surrogate model j surely reaches such a state, i.e.:

$$T^{i,j}_{\phi}(s) := P(\Phi_j(s) \mid \Phi_i(s))\,. \tag{3.12}$$

We compute this probability through the well-known *counting-worlds* technique (see [28]) under a classical interpretation of probability:

$$P(\Phi_j(s) \mid \Phi_i(s)) := \frac{\mid \Phi_j(s) \cap \Phi_i(s) \mid}{\mid \Phi_i(s) \mid}\,, \tag{3.13}$$

when $\Phi_i(s) \neq \emptyset$ and $P(\Phi_j(s) \mid \Phi_i(s)) = 1$, by convention, otherwise. Satisfiability of trustworthiness formulae is accordingly defined as follows.

Definition 58 (Satisfiability of trustworthiness formulae) *Given a TUS $\mathcal{M}_{TUS}$, a surrogate model $i \in \mathcal{A}$, a target model $j \in \mathcal{A}$, and a state $s \in \mathcal{S}$, the following condition holds:*

$$\mathcal{M}_{TUS}, s \models T^{i,j}_{\nabla h}\phi \;\text{ iff }\; P(\Phi_M(s) \mid \Phi_e(s))\nabla h\,. \tag{3.14}$$

3.5.6 Satisfiability of ϵ formulae

We interpret the degree of explanatory power as a combination of transparency, accuracy and trustworthiness. To effectively explain a property ϕ of a model $j \in \mathcal{A}$ to a user $u \in \mathbf{U}$, a surrogate model $i \in \mathcal{A}$ has to satisfy some conditions:

1. it has to be more transparent than the target-model j, i.e., $V^{i,u}_{\phi}(s) > V^{j,u}_{\phi}(s)$
2. it has to be sufficiently accurate in emulating ϕ;
3. it has to be sufficiently trustworthy in emulating ϕ.

Intuitively, if $V_\phi^{i,u}(s) \geq V_\phi^{j,u}(s)$, the explanation provided by i is not effective as the surrogate model is more (or equally) opaque than the model to be explained. In such cases, the degree of explanatory power is 0. In all other cases, the greater the difference between $V_\phi^{i,u}(s)$ and $V_\phi^{j,u}(s)$, the greater the degree of explanatory power of i, modulo i emulating j with a sufficient degree of accuracy and trustworthiness. In general, if a surrogate model i is highly transparent but not sufficiently accurate and trustworthy in emulating a property of the target model, then it will be not able to effectively explain it. Instead, the greater the respective degrees of accuracy and trustworthiness of a surrogate model i in emulating a property of the target model j, the greater the degree of explanatory power of i. Moreover, explanatory power is also connected with the explanatory relevance that each user attaches to, respectively, transparency, accuracy and trustworthiness. Some users, for example, prioritize accuracy over transparency and thus prefer an explanation based on surrogate models that are highly accurate and scarcely transparent rather than the opposite. Other users might prioritize transparency over accuracy, or accuracy over trustworthiness, and so on. In general, the contribution of transparency, accuracy and trustworthiness in determining the explanatory power of a surrogate model has to be weighted by the *explanatory relevance* that each user attaches to each of them. This can be obtained by weighting the respective degrees of transparency, accuracy and trustworthiness of a surrogate model with specific values that are provided by the explanatory relevance vectors $[x, y, z]^u$ for a given user $u \in \mathbf{U}$. By convention, we assume that $x + y + z = 1$ for each $[x, y, z]^u, u \in \mathbf{U}$. Each vector $[x, y, z]^u$ is thus analogous to a probability mass function defined over the space of explanatory relevance weights $\{x, y, z\}$.

Formally, we define the degree of explanatory power of a surrogate model $i \in \mathcal{A}$ against property ϕ of a model $j \in \mathcal{A}$ for user $u \in \mathbf{U}$ and with respect to initial state $s \in \mathcal{S}$, denoted by $Ex_\phi^{i,j,u}(s)$, as follows.

Definition 59 (Degree of explanatory power for single users)

$$Ex_\phi^{i,j,u}(s) := \begin{cases} 0 \;\; \text{iff} \;\; V_\phi^{i,u}(s) \leq V_\phi^{j,u}(s)\,, \\ x(V_\phi^{i,u}(s) - V_\phi^{j,u}(s)) + y(A_\phi^{i,j}(s)) + z(T_\phi^{i,j}(s)) \;\; \text{otherwise}\,. \end{cases} \tag{3.15}$$

In other words, if the surrogate model is less or equally transparent than the model to be explained, the degree of explanatory power is 0 by default; otherwise, such degree is computed as the weighted average among the surrogate model's degrees of accuracy, transparency, and the difference between the degrees of transparency of the surrogate model i and of the target model j, each value weighted for the explanatory relevance attributed to them by the user $u \in \mathbf{U}$.

We can thus introduce the satisfiability condition for single-user explanatory power formulae as follows.

Definition 60 (Satisfiability of single-user ϵ formulae) *Given a TUS $\mathcal{M}_{TUS}$, a surrogate model $i \in \mathcal{A}$, a target model $j \in \mathcal{A}$, a user $u \in \mathbf{U}$, and a state $s \in \mathcal{S}$, the following condition holds:*

$$\mathcal{M}_{TUS}, s \models Ex^{i,j,u}_{\nabla h}\phi \;\text{ iff }\; Ex^{i,j,u}_{\phi}(s)\nabla h\,. \tag{3.16}$$

This can be extended to groups of users $\Gamma \subseteq \mathcal{A}$, introducing the derivative notions of common and distributed degree of explanatory power. Let $[x, y, z]^{\Gamma}$ denote the vector of explanatory relevance for the groups of users $\Gamma \subseteq \mathbf{U}$. The elements of this vector denote the explanatory relevance that all the users $u \in \Gamma$ *jointly* attribute to, respectively, transparency, accuracy, and trustworthiness. These elements result from the application of a function $f(x^{u_1}, \ldots, x^{u_n}, y^{u_1}, \ldots, y^{u_n}, z^{u_1}, \ldots, z^{u_n})$ that maps the various elements of the vectors $[x, y, z]^u, u \in \Gamma$, to the respective elements of the vector $[x, y, z]^{\Gamma}$. In general, the choice of this function depends on the specific context of application. For practical reasons, here we assume that f corresponds to the *normalized average* of the respective elements of $[x, y, z]^u, u \in \Gamma$ that we can calculate as follows. First, we create a matrix whose rows are the various vectors $[x, y, z]^u, u \in \Gamma$. Second, we calculate the non-normalized vector $[X, Y, Z]^{\Gamma}$ by summing the three columns of the matrix. Finally, we normalize the vector by dividing both X, Y, and Z for their sum $X+Y+Z$, hence obtaining the normalized vector $[x, y, z]^{\Gamma}$. The normalized average seems to capture better than other functions the idea of a set of weights that provide the explanatory relevance that users $u \in \Gamma$ jointly attribute to the three properties of transparency, accuracy, and trustworthiness. Nevertheless, both the semantics and the model-checking procedures proposed in the following are independent from this choice and compatible with assuming other functions f in place of it. All that is needed is a feasible method to compute $[x, y, z]^{\Gamma}$ combining the various $[x, y, z]^u, u \in \Gamma$.

Once the joint explanatory relevance vector is obtained, specific definitions for common $CEx^{\Gamma,j,u}_{\phi}(s)$ and distributed $DEx^{\Gamma,j,u}_{\phi}(s)$ degrees of explanatory power can be introduced by analogy with that for single-user (see Definition 59), as follows:

Definition 61 (Degree of explanatory power for groups of users)

$$CEx^{i,j,u}_{\phi}(s) = \begin{cases} 0 \;\text{ iff }\; CV^{i,\Gamma}_{\phi}(s) \leq CV^{j,\Gamma}_{\phi}(s)\,, \\ x^{\Gamma}(CV^{i,\Gamma}_{\phi}(s) - CV^{j,\Gamma}_{\phi}(s)) + y^{\Gamma}(CA^{i,j}_{\phi}(s)) + z^{\Gamma}(CT^{i,j}_{\phi}(s)) \;\text{ else}\,, \end{cases} \tag{3.17}$$

$$DEx^{i,j,\Gamma}_{\phi}(s) = \begin{cases} 0 \;\text{ iff }\; DV^{i,\Gamma}_{\phi}(s) \leq DV^{j,\Gamma}_{\phi}(s)\,, \\ x^{\Gamma}(DV^{i,\Gamma}_{\phi}(s) - DV^{j,\Gamma}_{\phi}(s)) + y^{\Gamma}(DA^{i,j}_{\phi}(s)) + z^{\Gamma}(DT^{i,j}_{\phi}(s)) \;\text{ else}\,, \end{cases} \tag{3.18}$$

where x^{Γ}, y^{Γ}, and z^{Γ} denote the respective elements of the vector $[x, y, z]^{\Gamma}$. The satisfiability conditions are accordingly defined as follows.

Definition 62 (Satisfiability of group of users ϵ formulae) *Given a TUS $\mathcal{M}_{TUS}$, a surrogate model $i \in \mathcal{A}$, a target model $j \in \mathcal{A}$, a group of users $\Gamma \subseteq \mathbf{U}$, and a state $s \in \mathcal{S}$, the following conditions hold:*

$$\mathcal{M}_{TUS}, s \models CEx^{i,j,\Gamma}_{\nabla h}\phi \text{ iff } CEx^{i,j,\Gamma}_{\phi}(s)\nabla h\,, \tag{3.19}$$

$$\mathcal{M}_{TUS}, s \models DEx^{i,j,\Gamma}_{\nabla h}\phi \text{ iff } DEx^{i,j,\Gamma}_{\phi}(s)\nabla h\,. \tag{3.20}$$

3.6 Model-Checking

In this section we describe feasible procedures to model-check a given $\mathcal{M}_{TUS}$ against properties specified in the Ex-PCTL language. Let $\Lambda := \phi \mid \theta \mid \alpha \mid \epsilon$. Given a TUS $\mathcal{M}_{TUS}$, a state $s \in \mathcal{S}$, and a formula Λ, we want to define a procedure to check whether $\mathcal{M}_{TUS}, s \models \Lambda$. As for standard CTL and its extensions, the main procedure relies on the *parsing-tree* algorithm introduced in Section 2.6. Here, our task is thus to define a specific sub-routine to compute $Sat(\lambda)$ for the different kinds of Ex-PCTL formulae above introduced (i.e., ϕ, θ, α, and ϵ).

3.6.1 ϕ-formulae

We start by considering the case $\lambda := \phi$. Remeber that for ϕ-formulae the satisfiability conditions are defined with respect to a specific model $i \in \mathcal{A}$ in the multi-agent system. Consequently, when we apply the specific procedure to compute $Sat(\lambda)$ for $\lambda = \phi$, we have to specify the model $i \in \mathcal{A}$ included in the multi-agent system that we want to check against ϕ. By convention, we use notation $Sat_i(\phi)$ to highlight that we focus on model $i \in \mathcal{A}$. That said, the procedure to compute $Sat_i(\phi)$ is completely analogous to the one introduced in Section 2.7.3 for the computation of $Sat_i(\phi)$. That is, $Sat_i(\phi)$ is computed by an iterative application of the following recursive schema.

Definition 63 ($Sat_i(\phi)$)

$$\begin{aligned}
&Sat_i(\top) := \mathcal{S}\,,\\
&Sat_i(p) := \{s \in \mathcal{S} : p \in l^i(s)\}\,,\\
&Sat_i(\phi_1 \wedge \phi_2) := Sat_i(\phi_1) \cap Sat_i(\phi_2)\,,\\
&Sat_i(\neg\phi) := \mathcal{S} \setminus Sat_i(\phi)\,,\\
&Sat_i(P_{\nabla b}\psi) := \{s \in \mathcal{S} : P(i, s \models \psi)\nabla b\}\,,
\end{aligned}$$

where the specific procedures to compute $P(i, s \models \psi)$ for the various kinds of ψ-formulae are completely analogous the ones for the computation of $P(s \models \psi)$ introduced in Section 2.7.3.

3.6.2 θ-formulae

The specific procedure to compute $Sat(\theta)$ relies on the algorithm reported in Figure 9.The latter works as follows. It takes in input a TUS $\mathcal{M}_{TUS}$ and a formula θ. Given an $s \in \mathcal{S}$, it computes $\{\sim^u (s)\}$ for each $u \in \Gamma$ by checking for each $s' \in \mathcal{S}$ whether $s \sim^u s'$ or not (line 3). It then switches on the appropriate case, selected depending on the specification of θ (line 10). If $\theta := V^{i,u}_{\nabla h}\phi$ (line 11), then it computes the value of the ratio and check whether it satisfies the specified threshold ∇h; if the value satisfies the threshold, then the algorithm includes s in $Sat(\theta)$. If $\theta := CV^{i,\Gamma}_{\nabla h}\phi$ (line 16), it first calculates $\{\sim^{\Gamma}_{C} := it(\bigcup_{u\in\Gamma}\{\sim^u_D\}$, then it computes the value of the ratio and checks whether it satisfies the specified threshold ∇h; if the value satisfies the specified threshold, then the algorithm includes s in $Sat(\theta)$. Finally, if $\theta := DV^{i,\Gamma}_{\nabla h}\phi$ (line 22), it first calculates $\{\sim^{\Gamma}_{D}\} := \bigcap_{u\in\Gamma}\{\sim^u (s)\}$ and then proceeds as in the precedent case. The algorithm iterates the procedure for each $s \in \mathcal{S}$ and eventually output $Sat(\theta)$.

The overall time complexity of the algorithm is polynomial in $|\mathcal{S}|$. The computation of line 3 requires for each $u \in \Gamma$ and for each $s' \in \mathcal{S}$ to check whether $s \sim^u s'$, its execution, therefore, requires time linear in $\mid \mathcal{S} \mid$. The computations of the various ratios (lines 11, 16, and 22) require to compute few algebraic operations on sets and thus do not increase the time complexity which remains linear in $\mid \mathcal{S} \mid$. Accordingly, the time complexity of each iterative step is linear in $\mid \mathcal{S} \mid$. As the computation of $Sat(\theta)$ requires to iterate the procedure for each $s \in \mathcal{S}$, the overall time complexity results to be polynomial in $\mid \mathcal{S} \mid$.

3.6.3 α-formulae

The to compute $Sat(\alpha)$ relies on the algorithm reported in Figure 10. The latter works as follows. It takes in input a TUS $\mathcal{M}_{TUS}$ and a formula α (line 10). Given an $s \in \mathcal{S}$, it computes $\Phi_j(s)$ by checking for each $s' \in Sat_j(\phi)$ whether $h^j_{\{s'\}}(s) = 1$, i.e., whether s' is reachable from s according to the target-model j (line 4). Notice that, computing $h^j_{\{s'\}}(s)$ through Equation (1.8) at line 5 requires an amount of time polynomial in $|\mathcal{S}|$ for each $s' \in Sat_j(\phi)$. Since $Sat_j(s) \subseteq \mathcal{S}$, the time complexity of the overall procedure described in line 4 is polynomial in $|\mathcal{S}|$. Then the algorithm computes $\Phi_i(s)$ by checking for each $s' \in Sat_i(\phi)$ whether $h^i_{\{s'\}}(s)$ again through the procedure described in (1.8) (line 10). The time complexity of computing $h^i_{\{s'\}}(s)$ for each $s' \in Sat_i(\phi)$ is also polynomial in $|\mathcal{S}|$. To conclude, the algorithm checks whether $\mathcal{M}_{TUS}, s \models \alpha$ (line 15) by performing a simple sequence of algebraic operations on sets and thus checking whether the obtained result respects the threshold ∇h specified in the formula. Since this step consists of performing a very small number of simple algebraic operations on sets, it does not increase the time complexity of the procedure, which remains polynomial in $|\mathcal{S}|$. Finally, to obtain $Sat(\alpha)$, the algorithm iterates the procedure for each $s \in \mathcal{S}$. Consequently, the overall time

Algorithm 9: $Sat(\theta)$

Input: $\mathcal{M}_{TUS}$, θ
Output: $Sat(\theta)$

foreach $s \in \mathcal{S}$ **do**
  $Sat(\theta) \leftarrow \{\}$;
  foreach $u \in \Gamma$ **do**
    foreach $s' \in \mathcal{S}$ **do**
      $\{\sim^u(s)\} \leftarrow \{\}$ **if** $s \sim^u s'$ **then**
        $\{\sim^u(s) \leftarrow \{\sim^u(s)\} \cup \{s'\}$
      end
    end
  end
  switch θ **do**
    case $\theta := V^{i,u}_{\nabla h}\phi$ **do**
      if $\frac{(Sat_i(\phi) \cap \{\sim^u(s)\}}{Sat_i(\phi)})\nabla h$ **then**
        $Sat(\theta) \leftarrow Sat(\theta) \cup \{s\}$
      end
    end
    case $CV^{i,\Gamma}_{\nabla h}\phi$ **do**
      if $\frac{|Sat_i(s) \cap it(\bigcup_{u\in\Gamma}\{\sim^u(s)\}|}{|Sat_i(s)|}\nabla h$ **then**
        $Sat(\theta) \leftarrow Sat(\theta) \cup \{s\}$
      end
    end
    case $DV^{i,\Gamma}_{\nabla h}\phi$ **do**
      foreach $s \in \mathcal{S}$ **do**
        if $\frac{|Sat_i(s) \cap \bigcap_{u\in\Gamma}\{\sim^u(s)\}|}{|Sat_i(s)|}\nabla h$ **then**
          $Sat(\theta) \leftarrow Sat(\theta) \cup \{s\}$
        end
      end
    end
  end
end
return $Sat(\theta)$

Figure 3.4

complexity of the algorithm remains polynomial in $|\mathcal{S}|$.

3.6.4 ϵ-formulae

The procedure to compute $Sat(\epsilon)$ relies on the algorithm reported in Figure 11. The latter works as follows. It takes in input a TUS $\mathcal{M}_{TUS}$ and a formula α. It computes the joint vector of relevance $[x^\Gamma, y^\Gamma, z^\Gamma]$ as the element-wise product of the vectors of relevance $[x, y, x]^u$ for each $u \in \Gamma$. If $\Gamma := \{u\}$, it simply takes $[x, y, z]^u$. Given any $s \in \mathcal{S}$, it switches on the appropriate case depending on the specification of ϵ. If $\epsilon := Ex^{i,j,u}_{\nabla h}\phi$, it calculates $V^{i,u}_\phi(s)$, $V^{j,u}_\phi(s)$, $A^{i,j}_\phi(s)$ and $T^{i,j}_\phi(s)$ implementing the appropriate algorithms 9 and 10. Hence, it checks whether $V^{i,u}_\phi(s) \leq V^{j,u}_\phi(s)$, i.e., whether the degree of transparency of the surrogate model is less or equal then that of the target model. If this is the case, then the algorithm checks whether the threshold h specifying the desired degree of explanatory power is equal to 0. If $h = 0$, then the condition expressed in the semantics by Definition 60 is satisfied and the state s is included in $Sat(\epsilon)$. Otherwise the procedure terminates. Instead, if $V^{i,u}_\phi(s) > V^{j,u}_\phi(s)$, then the algorithm computes the value of the product $x(V^{i,u}_\phi(s) - V^{j,u}_\phi(s)) \cdot y(A^{i,j}_\phi(s)) \cdot z(T^{i,j}_\phi(s))$, i.e., the product among the respective degrees of accuracy and trustworthiness of the surrogate and the difference between the respective degree of transparency of the surrogate and the target model, all weighted with the respective values of the parameters included in the vector of relevance $[x, y, z]^u$. Finally, the algorithm checks whether the value of the product satisfies the specified threshold h and, if this is the case, it includes s in $Sat(\epsilon)$. The algorithm follows an analogous procedure when $\epsilon := CEx^{i,j,\Gamma}_{\nabla h}\phi$ or $\epsilon := DEx^{i,j,\Gamma}_{\nabla h}\phi$, with the only relevant difference that it computes $CV^{i,\Gamma}_\phi(s)$, $CV^{j,\Gamma}_\phi(s)$ (respectively, $DV^{i,\Gamma}_\phi(s)$ and $DV^{j\Gamma}_\phi(s)$) instead of $V^{i,u}_\phi(s)$ and $V^{j,u}_\phi(s)$. Finally, to compute $Sat(\epsilon)$, the algorithm iterates the procedure for each $s \in \mathcal{S}$.

The overall time complexity of the algorithm is clearly polynomial in the size of $\mathcal{S}$. In fact, as proved above, the time complexity of the procedures to compute $V^{i,u}_\phi(s)$, $V^{j,u}_\phi(s)$, $A^{i,j}_\phi(s)$ and $T^{i,j}_\phi(s)$ (or their respective multi-agent counterparts) are all polynomial in $|\ \mathcal{S}\ |$. The calculations of the various ratios consists of the execution of few algebraic operations on sets and, thus, do not increase the overall computational complexity, which remains polynomial in $|\ \mathcal{S}\ |$.

3.7 Example

3.7.1 Scenario 1

Let us consider as target-model a *probabilistic decision-tree* classifier whose task is to predict whether a given patient might develop schizophrenia based on the following Boolean

Algorithm 10: $Sat(\alpha)$

Input: $\mathcal{M}_{TUS}$, α
Output: $Sat(\alpha)$

$Sat(\alpha) \leftarrow \{\}$;
foreach $s \in \mathcal{S}$ **do**
  $\Phi_j(s) \leftarrow \{\}$;
  foreach $s' \in Sat_i(\phi)$ **do**
    if $h^j_{\{s'\}}(s) = 1$ **then**
      $\Phi_j(s) \leftarrow \Phi_i(s) \cup \{s'\}$
    end
  end
  $\Phi_i(s) \leftarrow \{\}$;
  foreach $s' \in Sat_i(\phi)$ **do**
    if $h^i_{\{s'\}}(s) = 1$ **then**
      $\Phi_i(s) \leftarrow \Phi_i(s) \cup \{s'\}$
    end
  end
  switch α **do**
    case $\alpha := A^{i,j}_{\nabla h}\phi$ **do**
      if $J(\Phi_i(s), \Phi_j(s))\nabla h$ **then**
        $Sat(\alpha) \leftarrow Sat(\alpha) \cup \{s\}$
      end
    end
    case $\alpha := T^{i,j}_{\nabla h}\phi$ **do**
      if $\frac{|\Phi_i(s) \cap \Phi_j(s)|}{|\Phi_i(s)|}\nabla h$ **then**
        $Sat(\alpha) \leftarrow Sat(\alpha) \cup \{s\}$
      end
    end
  end
end
return $Sat(\alpha)$

Figure 3.5

Algorithm 11: $Sat(\epsilon)$

Input: $\mathcal{M}_{TUS}$, ϵ
Output: $Sat(\epsilon)$

foreach $s \in \mathcal{S}$ **do**
  $Sat(\epsilon) \leftarrow \{\}$
  $V_{\phi}^{i,u}(s) \leftarrow \frac{|Sat_i(\phi) \cap \{\sim^u(s)\}|}{|Sat_i(\phi)|}$
  $CV_{\phi}^{i,u}(s) \leftarrow \frac{|Sat_i(\phi) \cap \{it(\bigcup_{u \in \Gamma}\{\sim^u(s)\}\}|}{|Sat_i(\phi)|}$
  $DV_{\phi}^{i,u}(s) \leftarrow \frac{|Sat_i(\phi) \cap \{\bigcap_{u \in \Gamma}\{\sim^u(s)\}\}|}{|Sat_i(\phi)|}$
  switch ϵ **do**
    case $\epsilon := Ex_{\nabla h}^{i,j,u}\phi$ **do**
      if $V_{\phi}^{i,u}(s) \leq V_{\phi}^{j,u}(s)$ **then**
        if $h = 0$ **then**
          $Sat(\epsilon) \leftarrow \{s\}$
        end
      end
      if $V_{\phi}^{i,u}(s) > V_{\phi}^{j,u}(s)$ **then**
        if $[x(V_{\phi}^{i,u}(s) - V_{\phi}^{j,u}(s)) \cdot y(A_{\phi}^{i,j}(s)) \cdot z(T_{\phi}^{i,j}(s))]\nabla h$ **then**
          $Sat(\epsilon) \leftarrow \{s\}$
        end
      end
    end
    case $\epsilon := CEx_{\nabla h}^{i,j,u}\phi$ **do**
      if $CV_{\phi}^{i,\Gamma}(s) \leq CV_{\phi}^{j,\Gamma}(s)$ **then**
        if $h = 0$ **then**
          $Sat(\epsilon) \leftarrow \{s\}$
        end
      end
      if $CV_{\phi}^{i,\Gamma}(s) > CV_{\phi}^{j,\Gamma}(s)$ **then**
        if $[x^{\Gamma}(CV_{\phi}^{i,\Gamma}(s) - CV_{\phi}^{j,\Gamma}(s)) \cdot y^{\Gamma}(CA_{\phi}^{i,\Gamma}(s)) \cdot z^{\Gamma}(CT_{\phi}^{i,\Gamma}(s))]\nabla h$ **then**
          $Sat(\epsilon) \leftarrow \{s\}$
        end
      end
    end
    case $\epsilon := Ex_{\nabla h}^{i,j,u}\phi$ **do**
      if $DV_{\phi}^{i,\Gamma}(s) \leq DV_{\phi}^{j,\Gamma}(s)$ **then**
        if $h = 0$ **then**
          $Sat(\epsilon) \leftarrow \{s\}$
        end
      end
      if $DV_{\phi}^{i,\Gamma}(s) > DV_{\phi}^{j,\Gamma}(s)$ **then**
        if $[x^{\Gamma}(DV_{\phi}^{i,\Gamma}(s) - DV_{\phi}^{j,\Gamma}(s)) \cdot y^{\Gamma}(DA_{\phi}^{i,\Gamma}(s)) \cdot z^{\Gamma}(DT_{\phi}^{i,\Gamma}(s))]\nabla h$ **then**
          $Sat(\epsilon) \leftarrow \{s\}$
        end
      end
    end
  end
end
return $Sat(\epsilon)$

state	l^j	l^i
0	m	m
1	m	m
2	g	g, d
3	d	d
4	m, g, d	m, g, d
5	m, g	m, g
6	m, d	d
7	g, d	g, d
8	m, g	m, g
9	p	p, g
10	p	p
11	m, d	m, d
12	m, g	m, g
13	g, d	g, d
14	p	p, m
15	p	p

Table 3.1: l^j

parameters: *gender*, *genetic pre-disposition*, and *presence of correlated psychiatric disorders*. We use labels m for "male", g for "presence of genetic disposition", d for "presence of psychiatric disorders" and p for "being schizophrenic".

The behaviour of the classifier can be described by a DTMC j defined over $\mathcal{S}$ and provided with: (i) a set of labels $AP^j := \{m, g, d, p\}$, (ii) a labelling function l^j and, (iii) a transition matrix T^j described in Table 3.2. There are sixteen different input/output states, i.e., $\mathcal{S} := \{s_0, s_1, \ldots, s_{15}\}$ stating the different profiles of the patients predictable given the analysis of the above parameters. The labelling of states of j according to l^j are reported in the first column of Table 3.1.

We use model j to predict the outcome for a patient whose actual profile is $\langle m, g, d, \neg p\rangle$, which corresponds to assuming s_4 as initial state (see Table 3.1). We know that for input $\langle m, g, d, \neg p\rangle$ the probability that the model predicts p, i.e., "schizophrenia", is equals to 1, which means that formula $\phi := P_{=1}\top \bigcup(p)$ holds in s_4.

We are interested in explaining the behaviour of j locally on input $\langle m, g, d, \neg p\rangle$. To this aim, we build a surrogate model j (e.g. a rule-based system)[18] able to emulate the behaviour of the classifier locally on s_4. The stochastic behaviour of i is described by

[18]Remember that a surrogate model is an agent able to (locally) emulate the behaviour of the target-model and usually considered more transparent than the latter.

another labelled DTMC defined over the same set of states $\mathcal{S}$ and whose characteristic labelling function l^i and transition matrix T^i are reported, respectively, in the second column of Table 3.1 and in Table 3.3.

To complete the scenario, we consider a user u_1 (e.g., a physician) interacting with both model j and model i. The epistemic access abilities of u_1 are described by the relation $\sim^{u_1}$. According to the specification of $\sim^{u_1}$, we have that $\sim^{u_1}(s_4) = \{s_0, s_1, s_2, s_3, s_4, s_6, s_7, s_{10}, s_{11}, s_{12}, s_{13}\}$.[19] The explanatory relevance vector $[x, y, z]^{u_1}$ characteristic of u_1, instead, is specified as $[0.1, 0.5, 0.4]^{u_1}$.

In the above specified scenario, we are interested in checking whether the explanatory power of i against property $\phi := P_{=1}\top\bigcup(p)$ of the target-model j for user u_1 locally computed on input $\langle m, g, d, \neg p\rangle$ is higher than a given desirable threshold, for instance ≥ 0.4. This corresponds to check whether:

$$\mathcal{M}_{TUS}, s_4 \models Ex^{i,j,u}_{\geq 0.4} P_{=0.1}\top\bigcup(p) \tag{3.21}$$

For executing the task, we apply the procedure described in algorithm 11. First, we calculate $Sat_j(P_{=1}\top\bigcup(p))$ and $Sat_i(P_{=1}\top\bigcup(p))$ through the procedure described in Section 3.6.1, obtaining:

$$Sat_i(P_{=1}\top\bigcup(p)) = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}, s_{15}\},$$
$$Sat_j(P_{=1}\top\bigcup(p)) = \{s_1, s_4, s_5, s_8, s_9, s_{10}, s_{14}, s_{15}\},$$

Second, we calculate:

$$\sim^{u_1}(s_4) \cap Sat_i(P_{=1}\top\bigcup(p)) = \{s_0, s_1, s_2, s_3, s_4, s_6, s_7, s_{10}, s_{11}, s_{12}, s_{13}\}$$
$$\sim^{u_1}(s_4) \cap Sat_j(P_{=1}\top\bigcup(p)) = \{s_1, s_4, s_{10}\},$$

Third, we calculate:

$$V^{i,u_1}_{P_{=1}\top\bigcup(p)}(s_4) = \frac{|\ Sat_i(P_{=1}\top\bigcup(p)) \cap \{\sim^{u_1}(s_4)\}\ |}{Sat_i(P_{=1}\top\bigcup(p))} = \frac{11}{16} = 0.6875$$
$$V^{j,u_1}_{P_{=1}\top\bigcup(p)}(s_4) = \frac{|\ Sat_j(P_{=1}\top\bigcup(p)) \cap \{\sim^{u_1}(s_4)\}\ |}{Sat_j(P_{=1}\top\bigcup(p))} = \frac{3}{8} = 0.375$$

[19] Remember that $\sim^{u_1}(s_4)$ denotes the set of states that are epistemically accessible from s_4 by u_1.

and, thus:

$$V^{i,u_1}_{P_{=1}\top\bigcup(p)}(s_4) - V^{j,u_1}_{P_{=1}\top\bigcup(p)}(s_4) = 0.3125\,,$$
$$A^{i,j}_{P_{=1}\top\bigcup(p)}(s_4) = 0.4286\,,$$
$$T^{i,j}_{\phi}(s_4) = \frac{Sat_j(P_{=1}\top\bigcup(p)) \cap Sat_i(P_{=1}\top\bigcup(p))}{Sat_i(P_{=1}\top\bigcup(p))} = \frac{8}{16} = 0.5\,,$$

Finally, given $\phi = P_{=1}\top\bigcup(p)$, we can calculate:

$$Ex^{i,j,u_1}_{\phi}(s_4) = 0.1 * (V^{i,u_1}_{\phi)}(s_4) - V^{j,u}_{\phi}(s_4)) + 0.5 * A^{i,j}_{\phi}(s_4) + 0.4 * T^{i,j}_{\phi}(s_4) = 0.44555$$

Hence, as the obtained degree of explanatory power is greater than the specified threshold, we can conclude that the formula specified in (3.21) holds in the model and, thus, that the explanatory power of i against property $P_{=1}\top\bigcup(p)$ of the target-model j is sufficient for the purposes of the specified user.

3.7.2 Scenario 2

Suppose now that surrogate model i has been conceived to explain j not only to a specific class of users, but to all the potential different users of j. Let us assume that there exist three different kinds of users for model j that we denote by u_1, u_2, and u_3. According to the specifications of $\sim^{u_2}$ and $\sim^{u_3}$, we have that:

$$\sim^{u_2}(s_4) = \{s_0, s_1, s_2, s_3, s_4, s_6, s_7, s_{10}, s_{11}, s_{12}, s_{13}\}\,,$$
$$\sim^{u_3}(s_4) = \{s_4, s_6, s_7, s_{10}, s_{11}, s_{12}, s_{13}\}$$

Furthermore, we have that:

$$[x, y, z]^{u_2} = [0.6, 0.1, 0.3]\,,$$
$$[x, y, z]^{u_3} = [0.6, 0.2, 0.2]$$

We are interested in checking whether the explanatory power of surrogate model i satisfies the desirable threshold specified above (i.e., ≥ 0.4) also in case of a group Γ including all the possible potential users of the target-model j, i.e., in case $\Gamma = \{u_1, u_2, u_3\}$. We consider two different conditions:

$$\mathcal{M}_{TUS}, s_4 \models CEx^{i,j,\Gamma}_{\geq 0.4} P_{=0.1} \top \bigcup (p)\,,$$
$$\mathcal{M}_{TUS}, s_4 \models DEx^{i,j,\Gamma}_{\geq 0.4} P_{=0.1} \top \bigcup (p)$$

The first condition refers to the *common* degree of explanatory power in the group of users Γ while the second refers to the *distributed* degree of explanatory power. In the former case, we consider how much the explanation provided by the surrogate model is effective for users in Γ on the basis of the *common knowledge* of the target-system that they possess. In the second case, instead, we consider effectiveness on the basis of the *distributed knowledge* [20].

To check whether the conditions expressed above are satisfied we follow the algorithm reported in (11). First, we calculate:

$$\sim^{\Gamma}_{C} (s_4) = it(\bigcup_{u \in \Gamma} \sim^{u} (s_4)) = \{s_0, s_1, s_2, s_3, s_4, s_6, s_7, s_{10}, s_{11}, s_{12}, s_{13}\}\,,$$
$$\sim^{\Gamma}_{D} (s_4) = \bigcap_{u \in \Gamma} \sim^{u} (s_4) = \{s_4, s_6, s_7, s_{10}, s_{11}, s_{12}, s_{13}\}$$

Second, we calculate:

$$\sim^{\Gamma}_{C} (s_4) \cap Sat_i(P_{=1} \top \bigcup (p)) = \{s_0, s_1, s_2, s_3, s_4, s_6, s_7, s_{10}, s_{11}, s_{12}, s_{13}\}\,,$$
$$\sim^{\Gamma}_{C} (s_4) \cap Sat_j(P_{=1} \top \bigcup (p)) = \{s_1, s_4, s_{10}\}$$

and

$$\sim^{\Gamma}_{D} (s_4) \cap Sat_i(P_{=1} \top \bigcup (p)) = \{s_4, s_6, s_7, s_{10}, s_{11}, s_{12}, s_{13}\}\,,$$
$$\sim^{\Gamma}_{D} (s_4) \cap Sat_j(P_{=1} \top \bigcup (p)) = \{s_4, s_{10}\}$$

Third, we calculate:

$$CV^{i,\Gamma}_{P_{=1} \top \bigcup (p)} - CV^{j,\Gamma}_{P_{=1} \top \bigcup (p)} = 0.3125$$
$$DV^{i,\Gamma}_{P_{=1} \top \bigcup (p)} - DV^{j,\Gamma}_{P_{=1} \top \bigcup (p)} = 0.125$$

[20] In epistemic logic, common knowledge usually refers to the information shared by all the subjects in a group, while *distributed knowledge* refers to the information that can be derived from that possessed by each single subject in the group.

Table 3.2: T^j

0.25	0.25	0.25	0.25	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0.5	0.5	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0.25	0	0	0.25	0.25	0.25	0	0	0	0	0	0	0
0	0	0	0	0.5	0	0	0	0	0.5	0	0	0	0	0	0
0	0	0	0	0	0.5	0	0	0	0	0.5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0
0	0	0	0	0	0	0	0.5	0	0	0	0	0	0.5	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5
0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0
0	0	0	0	0	0	0.5	0	0	0	0	0	0.5	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5

Fourth, we calculate the distributed explanatory relevance vector $[x, y, z]^\Gamma = [0.38, 0.24, 0.38]$, obtained as the normalized average of the explanatory relevance vectors of the single-users $u \in \Gamma$.

Finally, given $\phi = P_{=1} \top \bigcup(p)$, we can calculate:

$$\begin{aligned} CEx_\phi^{i,j,\Gamma}(s_4) &= 0.38 * (CV_{\phi)}^{i,\Gamma}(s_4) - CV_\phi^{j,\Gamma}(s_4)) + 0.24 * A_\phi^{i,j}(s_4) + 0.38 * T_\phi^{i,j}(s_4) \\ &= 0.4125 \\ DEx_\phi^{i,j,\Gamma}(s_4) &= 0.38 * (DV_\phi^{i,\Gamma}(s_4) - DV_\phi^{j,\Gamma}(s_4)) + 0.24 * A_\phi^{i,j}(s_4) + 0.38 * T_\phi^{i,j}(s_4) \\ &= 0.2025 \end{aligned}$$

In the first case, we see that $CEx^{i,j,\Gamma}_{P_{=1}\top\bigcup(p)}(s_4) \geq 0.4$ and, thus, we can conclude that formula $CEx^{i,j,\Gamma}_{\geq 0.4} P_{=0.1} \top \bigcup(p)$ holds in the specified model and, thus, that the common degree of explanatory power of i against property $P_{=1} \top \bigcup(p)$ of the target-model j according to users in Γ is sufficient. In the second case, instead, we see that $DEx^{i,j,\Gamma}_{P_{=1}\top\bigcup(p)}(s_4) \leq 0.4$. Consequently, we can conclude that formula $DEx^{i,j,\Gamma}_{\geq 0.4} P_{=0.1} \top \bigcup(p)$ does not hold for state s_4 of the specified model and, thus, that the distributed degree of explanatory power of i against property $P_{=1} \top \bigcup(p)$ of the target-model j according to users in Γ is not sufficient.

3.8 Conclusions

In this chapter, we presented a framework to model and check the explanatory power of surrogate models used in XAI to post-hoc explain the behaviour of opaque systems. Explanatory power is defined as a function of three more fundamental properties, which are

Table 3.3: T^i

0.25	0.25	0.25	0.25	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0.5	0.5	0	0	0	0	0	0	0	0	0	0
0	0	0.5	0	0	0	0.5	0	0	0	0	0	0	0	0	0
0	0	0	0.25	0	0	0.25	0.25	0.25	0	0	0	0	0	0	0
0	0	0	0	0.5	0	0	0	0	0.5	0	0	0	0	0	0
0	0	0	0	0	0.5	0	0	0	0	0.5	0	0	0	0	0
0	0	0.5	0	0	0	0	0	0	0	0	0.25	0.25	0	0	0
0	0	0	0	0	0	0	0.5	0	0	0	0	0	0.5	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5
0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0	0
0	0	0	0	0	0	0.5	0	0	0	0	0	0.5	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.5

transparency, *accuracy*, and *trustworthiness*. We introduced Ex-PCTL, a new language to specify explanatory power along with its fundamental properties, equipped with a proper semantics based on a multi-agent structure that includes both models and users. This semantics treatment follows from the widely shared conviction, among XAI practitioners, that whether and to what extent a model is opaque and needs an explanation also depends on who are its users. Finally, we present specific algorithms to model-check the transparency, accuracy, trustworthiness, and explanatory power of surrogate models in time polynomial in the number of their possible alternative states.

Further possible developments include the extension of the proposed semantics to AI systems whose behaviour cannot be described through DTMCs but requires, for instance, continuous-time Markov chains, Markov decision processes or hidden Markov models.

Chapter 4

Probabilistic Model Checking with Imprecise Probabilities

Abstract

This chapter presents a novel framework for model checking stochastic multi-agent systems based on the theory of *imprecise probabilities* and the related imprecise Markov models. The advantage of the latter is that they allow for modeling non-stationary and not fully knowledgeable stochastic systems without incurring computational complexity issues, as the results in the chapter aim to prove. The chapter begins with a detailed state of the art of imprecise Markov models and the existing methods to compute probabilistic inferences over them. It then introduces Imprecise Probabilistic Interpreted Systems (IPIS), a new class of imprecise Markov models to represent stochastic multi-agent systems, and their extension with rewards (IPIRS). Hence, it develops algorithms to compute probabilistic inferences over IPISs and IPIRSs in time polynomial to the models' number of states. Subsequently, the chapter introduces a new logic and relative model-checking algorithms to verify properties of non-stationary stochastic multi-agent systems using IPISs and IPIRSs. As the main result, the chapter proves that shifting from precise to imprecise Markov models does not affect the computational complexity of the main model checking tasks, which remains time-polynomial in the target-systems' number of states. The chapter is built upon preliminary results published in [157, 158] and [159].

4.1 Introduction

Despite its success, probabilistic model checking suffers from the well-known limitation of requiring all the transition probabilities to be defined by sharp, "precisely specified", numerical values. This constraint might be critical in several applications, as it prevents from modelling both non-stationary systems and systems characterised by a partial uncertainty on the values of transition probabilities. An appealing way to overcome this limitation is represented by so-called parametric Markovian models [39], where precise state transition

probabilities are replaced with unknown parameters. This is the solution adopted, for instance, by [9]. However, complexity issues related with the corresponding model checking procedure, based on fraction-free Gaussian elimination, limit its applicability only to models of small size.

A less explored alternative is provided by the formalism of imprecise probabilities [173] and related imprecise Markov models, notably *imprecise Markov chains* (IMC) and their extensions [163, 48, 97, 157]. These can be seen as the imprecise counterparts of standard Markov chains and are obtained by replacing single-valued probability distributions with so-called *credal sets*, i.e., sets of probability distributions describing the model and compatible with given specific constraints [40].

A first attempt to extend probabilistic model checking to the framework of imprecise probabilities has been proposed in [163]. This paper introduces an imprecise version of PCTL based on IMCs. It shows that shifting from precise to imprecise Markov chains does not increase the time complexity of the relevant model checking tasks, which remain polynomial in the number of states of the models. An extension of the results in [163] is provided in [157], where both a semantics and the relative model checking procedures for imprecise Markov reward models are introduced.

In both [163] and [157] only single-agent systems are considered. A first multi-agent extension is presented in [158]. In this paper, the authors introduce a class of structures to model epistemic-stochastic multi-agent systems called *imprecise probabilistic interpreted systems*, a related language called EIPCTL (Epistemic Imprecise PCTL), and relative model checking procedure.

The present chapter generalises and extends the framework and the results in [157] and [158]. In particular, it introduces an imprecise-probabilistic framework to model and verify multi-agent systems with rewards. The language of EIPCTL is accordingly extended to specify reward properties and feasible procedures for model checking imprecise probabilistic interpreted systems with reward. The chapter also extends the preliminary results outlined in [163] and [158], hence proving that shifting from precise to imprecise models does not increase the overall time complexity of the relevant model checking tasks, which remains polynomial in the number of states of the models. The developed framework is finally tested on a case-study borrowed from the medical domain.

4.2 Imprecise Markov Models

In this section we provide imprecise-probabilistic counterparts for the Markov models presented in Chapter 1. We also show how the efficient inference algorithms for standard Markov models described in Chapter 1 can be easily extended to the imprecise probabilities framework without increased computational costs. These results partially rely on recent

works about imprecise Markov models [97, 161].

In introducing imprecise Markov models, we follow the so-called *measure-theoretic* interpretation and relies on the formalism of *credal sets* [116]. The alternative *game-theoretic* formalisation [148] is briefly mentioned without going into details insofar the two formalisms are equivalent for the inference tasks relevant for this work, as proved in [165].

4.2.1 Imprecise Transition Matrices

Given a variable S, a *Credal Set* (CS) $K(S)$ is a set of probability mass functions over S. The upper expectation of a real-valued function f of S with respect to CS $K(S)$ is intended as $\overline{E}[f(S)] := \sup_{P(S)\in K(S)} \sum_{s\in\mathcal{S}} f(s)\cdot P(s)$ (the lower expectation $\underline{E}[f(S)]$ is analogously defined). Here we only consider closed and convex CSs induced by a finite number of linear constraints. These are polytopes in the probability simplex with a finite number of extreme points collected in a set $Ext[K(S)]$. For these CSs, upper (lower) expectations can be equivalently obtained by taking the maximum (minimum) with respect to the precise expectations computed on the extreme points. Conditional CSs might be defined analogously [42].

In this framework, an *imprecise* transition matrix $\mathcal{T}$ is defined as a collection of conditional CSs $\{K(S'|s)\}_{s\in\mathcal{S}}$, each one representing a separately specified row of the matrix. This allows for defining precise transition matrices whose rows are obtained by taking a $P(S'|s) \in K(S'|s)$ for each $s \in \mathcal{S}$. Each one of these matrices represents a stochastic behaviour compatible with the "imprecise" specification given by $\mathcal{T}$.

4.2.2 Imprecise Markov Chains

As a first example of imprecise Markov model, we consider (discrete-time) *imprecise Markov chains* (IMCs). Note that there exist two main ways of formalising IMCs in the literature. On the one hand, the *measure-theoretic* characterisation defines an IMC as a family of (discrete-time) Markov models compatible with beliefs about initial and transition probabilities. On the other hand, the *game-theoretic* characterisation is grounded on the game-theoretic view of probability popularised in [148] that, applied to the theory of stochastic processes directly leads to imprecise models[1]. The two characterisations are different but have been recently proved to coincide for all expectations on the following domains: (i) monotone pointwise limits of finitary real-valued functions, and (ii) bounded below Borel-measurable variables [165]. In this work we focus on measure-theoretic IMCs only. However, all the inferences we consider fall under (i) and thus, for the purposes of our work, the two characterisations can be considered equivalent.

[1]See [41, 50] for an in-depth discussion on the relation between measure-theoretic and game-theoretic IMCs.

Given a CS $K(S_0)$ and an imprecise transition matrix $\mathcal{T}$, both defined over $\mathcal{S}$, the (discrete-time) IMC $\mathcal{M}$ induced by $K(S_0)$ and $\mathcal{T}$ can be defined as the largest set of (discrete-time) stochastic processes that are *compatible* with $K(S_0)$ and $\mathcal{T}$.

The term "compatible" here deserves an exact characterisation. In the imprecise probabilities literature, indeed there exist at least two different criteria for establishing compatibility, which depends on the imprecise interpretation of the notion of stochastic irrelevance, and, consequently, of the Markov property one considers [80]. The two notions of irrelevance typically involved for IMCs are *strong independence* and *epistemic irrelevance*. The former is defined via product-independence of the CS extreme points: we say that $K(S)$ and $K(S')$ are strong-independent if and only if, for all $P(S) \in Ext[K(S)]$ and all $P(S') \in Ext[K(S')]$, it holds that $P(S, S') = P(S) \cdot P(S')$. The latter is defined via conditioning: we say that $K(S)$ is epistemically irrelevant for $K(S')$ if and only if $K(S' \mid s) = K(S')$ for each $s \in \mathcal{S}$.[2] Notice that, unlike strong independence, epistemic irrelevance is asymmetric, i.e., the irrelevance of $K(S)$ for $K(S')$ does not entail the irrelevance of $K(S')$ for $K(S)$.

Following [96] and [161], and also the early work of [163], in this paper we focus on epistemic irrelevance. Notably, by exploiting the results in [116], the imprecise-probabilistic inferences considered in the rest of this paper can easily proved to be independent of the specific characterisation we adopt. Epistemic irrelevance leads to an imprecise characterisation of the Markov property practically corresponding to assume that "whenever the agent knows the current state, then her beliefs about future states are not altered upon learning what states were visited in the past" [80, p.265]. A formal definition of IMC can thus be given as follows:[3]

Definition 64 (Imprecise Markov chain under epistemic irrelevance) *Given $K(S_0)$ and $\mathcal{T}$, an IMC $\mathcal{M}$ (under epistemic irrelevance) is defined as the largest set of all, potentially non-Markov, non-homogeneous, stochastic processes for which, for all $t \in \mathbb{N}$ and all $s_0, \ldots, s_t \in \mathcal{S}^t$, there is some $T \in \mathcal{T}$ such that $P(S_{t+1} = s' \mid S_{0:t} = s_{0:t}) = T(s_{0:t}, s')$ for all $s' \in \mathcal{S}$.*

Furthermore, each IMC is uniquely identified by a set of probability measures $P^M : \sigma(\Pi) \to [0, 1]$ that we denote by $K^{\mathcal{M}}$. Each $P^M \in K^{\mathcal{M}}$ uniquely identifies a (potentially non-Markov and non-homogeneous) stochastic process compatible with the IMC identified by $K^{\mathcal{M}}$. The IMC identified by $K^{\mathcal{M}}$ is also uniquely identified by $K_s^{\mathcal{M}}$, which is the credal set including all the *conditional* probability distribution P_s^M obtained by conditioning the various $P^M \in K^{\mathcal{M}}$ on a given initial state $s \in \mathcal{S}$. As detailed in the next section, inferences in IMCs are consequently intended as the computation of lower and upper expectations with respect to such credal set.

[2] For a more detailed characterisation of the difference between these notions, we refer to [119].

[3] The definition here reported is that introduced in [161]. An analogous definition is given in [97].

Before to move on, notice that, as in the precise case, we are interested here in *labelled* IMCs, i.e., IMCs augmented with a finite set of atomic propositions AP and a labelling function $l : \mathcal{S} \to 2^{AP}$. In what follows, when using the term IMC, we always refer to their labelled extensions.

4.2.3 Inference in Imprecise Markov Chains

To compute inferences in IMCs, let us first introduce the analogous of the transition operator in Equation (1.4). This is obtained by taking the bounds with respect to all the possible (precise) specifications of transition probabilities consistent with the imprecise transition matrix of the IMC. For upper bounds, this corresponds to the following non-linear upper operator:

$$(\overline{\mathcal{T}} f)(s) := \max_{T(s,S') \in \mathcal{T}(s,S')} \sum_{s' \in \mathcal{S}} T(s, s') \cdot f(s') \,, \tag{4.1}$$

while an analogous definition, with the minimum replacing the maximum, holds for the lower operator $\underline{\mathcal{T}}$ [161, Eq. 1]. Equation (4.1) can be computed by solving $|\mathcal{S}|$ linear programming tasks whose feasible regions are the conditional CSs in the definition of $\mathcal{T}$. This is possible, in particular, because we assume (Sec. 4.2.1) that each row of $\mathcal{T}$ is separately specified and consists of a conditional CS $K(S' \mid s)$ described by a finite number of linear constraints.

An iterated application of the above operators can be used to compute the bounds of the probability of reaching a given set of states after a number of time steps t, as shown by the following result.

Theorem 11 *Given an event $B \subseteq \mathcal{S}$ and a time $t \in \mathbb{N}$, let $\overline{P}_s(s_t \in B)$ denote the upper bound for the probability of reaching B after t time steps when starting from s. It holds that:*

$$\overline{P}_s(s_t \in B) = \overline{\mathcal{T}}^t \mathbb{I}_B(s) \,. \tag{4.2}$$

A similar result allows for obtaining the lower probability by means of the lower operator. For the sake of conciseness, in the rest of the paper, we only report the results for upper probabilities, expectations and transition operators. The lower bounds can always be obtained by replacing the upper transition operator with its lower analogous.

For what concerns the upper hitting probability, the latter can be regarded as the upper bound of the h_B defined in Equation (10) with respect to the set $K^{\mathcal{M}}$, as detailed in the following definition.

Definition 65 (Upper hitting probability) *Given a IMC $\mathcal{M}$, a set of states $B \subseteq \mathcal{S}$, and an initial state $s \in \mathcal{S}$:*

$$h_B(s) := \max_{P_s^M \in K^{\mathcal{M}}} \sum_{\substack{\hat{\pi} \in \mathrm{Paths}_{fin}(s) : \exists t \in \mathbb{N} \text{ s.t.} \\ \hat{\pi}(t) \in B \wedge \forall \tau < t, \hat{\pi}(\tau) \notin B}} P_s^M(Cyl(\hat{\pi}))$$

The latter can be compute as the minimal solution of the following system of equations [97, Corollary 19]:[4]

$$\overline{h}_B = \mathbb{I}_B + \mathbb{I}_{B^c} \overline{\mathcal{T}}\, \overline{h}_B \,. \tag{4.3}$$

Differently from the precise case, the system in Equation (4.3) is non-linear and cannot be solved by the standard methods typically used in the precise case. Nevertheless, as we show below, it is possible to apply a schema analogous to that in Equation (1.8) and compute $\overline{h}_B$ by recursion over increasing values of t (see, [97]). Let $\overline{h}_B^t$ denote the upper hitting probability of B for a finite number of time-steps $t \in \mathbb{N}$ conditional on $S_0 = s$. For $t = 0$, we trivially have that $\overline{h}_B^{t=0} = \mathbb{I}_B$. For $t > 0$, we have instead the following recursion:

$$\overline{h}_B^t = \mathbb{I}_B + \mathbb{I}_{B^c} \overline{\mathcal{T}}\, \overline{h}_B^{t-1}. \tag{4.4}$$

In practice, the procedure consists of t iterated applications of the transition operator $\overline{\mathcal{T}}$ and, consequently, requires the solution of $|\mathcal{S}| \cdot t$ linear programming tasks. The time complexity of the procedure is therefore polynomial in $|\mathcal{S}| \cdot t$, exactly as in the precise case.

As for standard DTMCs, it is proved that the least fixed point of Equation (4.4) is the minimal non-negative solution of the schema in Equation (4.3) (see, [97, Prop 16]). We can thus compute $\overline{h}_B$ simply by iterating the schema in Equation (4.4) over increasing values of t until converge. The overall time complexity results polynomial in $|\mathcal{S}|t^*$ as in the precise case. Each iteration step is based on a one-step application of the upper transition operator $\overline{\mathcal{T}}$ and requires the solution of $|\mathcal{S}|$ linear optimisation tasks: its time-complexity is therefore polynomial in $|\mathcal{S}|$. As t^* further iterations are necessary to reach convergence, the overall time-complexity results polynomial in $|\mathcal{S}|t^*$.

4.2.4 Imprecise Markov Reward Models

The *imprecise Markov reward models* (IMRMs) are the first extension of IMCs we consider. The latter can be defined as the imprecise counterpart of a MRM and consists of a pair $\langle \mathcal{M}, rew \rangle$ of a IMC $\mathcal{M}$ and a reward function $rew : \mathcal{S} \to \mathbb{R}$. For IMRMs, we characterise the expected cumulative reward $ExpRew_B$ by its upper and lower bounds, respectively

[4] See also [161].

denoted $\overline{Exp}Rew_B$ and $\underline{Exp}Rew_B$. As in the precise case, we restrict the latter to only $s \in \mathcal{S}^B_{=1}$, where $\mathcal{S}^B_{=1}$ is now defined as the set of all $s \in \mathcal{S}$ such that $\underline{h}_B(s) = 1$.

Given an event $B \subseteq \mathcal{S}$ and a path $\pi \in \Pi^{M_{\text{DTIMC}}}$, let us consider the cumulative reward $Rew_B(\pi)$ earned along π until visiting an $s \in B$ for the first time, as in Def. 12. The *upper* $\overline{Exp}Rew_B(s)$ expected cumulative rewards earned until reaching B starting from $s \in \mathcal{S}$ can be defined as the upper expectation of Rew_B conditional on the initial state $s \in \mathcal{S}$, i.e., $\overline{Exp}Rew_B(s) \coloneqq \overline{E}[Rew_B|s]$.

Inspired by Theorem 4 valid for the precise case, we introduce an imprecise version of the recursive scheme presented in Equation (1.10). To this end, let $\overline{Exp}Rew^0_B(s) \coloneqq rew(s)$ for every $s \in \mathcal{S}^B_{=1}$. Instead, for each $t \in \mathbb{N},\ t \neq 0$, let $\overline{Exp}Rew^t_B(s)$ be defined as follows:

$$\overline{Exp}Rew^t_B(s) \coloneqq \begin{cases} rew(s) & \text{if } s \in B\,, \\ rew(s) + (\overline{T}\,\overline{Exp}Rew^{t-1}_B)(s) & \text{otherwise,} \end{cases} \tag{4.5}$$

Similarly to the precise case, $\overline{Exp}Rew^t_B$ can be given a clear interpretation via the following theorem.

Theorem 12 *For every $t \in \mathbb{N}$, it holds that*

$$(\forall s \in S^B_{=1})\ \overline{Exp}Rew^t_B(s) = \overline{E}[Rew^t_B \mid S_0 = s], \tag{4.6}$$

where for each $\pi \in \Pi^{M_{\text{DTIMC}}}$, $Rew^0_B(\pi) \coloneqq rew(\pi(0))$, and for each $t \in \mathbb{N},\ t \neq 0$,

$$Rew^t_B(\pi) \coloneqq \begin{cases} Rew_B(\pi) & \text{if } \exists t^* \leq t : (\forall \tau < t^*)\ \pi(\tau) \notin B, \pi(t^*) \in B, \\ \sum_{\tau=0}^{t} rew(\pi(\tau)) & \text{otherwise.} \end{cases} \tag{4.7}$$

Proof 6 *The proof is analogous to the one of [97, Lemma 9] given for a similar result. Notice that results and lemmas from [97] that we will use in the following are originally formulated for game-theoretic IMCs. As from results proved in [165], for the domain of functions we consider in what follows, the measure-theoretic and the game-theoretic formulations can be considered equivalent. We give the proof for $\overline{Exp}Rew^t_B$, the one for $\underline{Exp}Rew^t_B$ will follow an analogous reasoning.*

The statement is proven by induction and we first prove the induction base. To this end, note that Rew^0_B only depends on the state S_0, so, it holds that

$$(\forall s \in S^B_{=1})\ \overline{E}[Rew^0_B \mid S_0 = s] = rew(s) =: \overline{Exp}Rew^0_B(s). \tag{4.8}$$

Next, as in the proof of Theorem 3, we interpret Rew^t_B as a function on S^{t+1}. Moreover, for any $t \in \mathbb{N},\ t \neq 0$ and $s_0, ..., s_t \in S^{t+1}$ we rewrite $Rew^t_B([s_i]^t_{i=0})$ as:

$$Rew^t_B([s_i]^t_{i=0}) = rew(s_0) + I_{B^c}(s_0) Rew^{t-1}_B([s_i]^t_{i=1}), \tag{4.9}$$

with $Rew_B^0(s_0) = rew(s_0)$. *Exploiting this observation, we can proceed with the induction step. Assuming that the statement is true for* $t-1$, *with* $t \in \mathbb{N}$, $t \neq 0$, *we prove that it is also true for* t. *First, for every* $s \in S_{=1}^B$, *we have that*

$$\begin{aligned}
&\overline{E}[Rew_B^t([S_i]_{i=0}^t) \mid S_0 = s] = \\
&\quad = \overline{E}[rew(s) + I_{B^c}(s) Rew_B^{t-1}([S_i]_{i=1}^t) \mid S_0 = s] = \\
&\quad = rew(s) + I_{B^c}(s)\overline{E}[Rew_B^{t-1}([S_i]_{i=1}^t) \mid S_0 = s] = \\
&\quad = rew(s) + I_{B^c}(s)\overline{E}[\overline{E}[Rew_B^{t-1}([S_i]_{i=1}^t) \mid [S_i]_{i=0}^1] \mid S_0 = s],
\end{aligned}$$

where the last step is based on [97, Proposition 38].

Now, $\overline{E}[Rew_B^{t-1}([S_i]_{i=1}^t) \mid [S_i]_{i=0}^1]$ *does not depend on the initial state* S_0, *hence*

$$\begin{aligned}
\overline{E}[Rew_B^{t-1}([S_i]_{i=1}^t) \mid [S_i]_{i=0}^1] &= \\
&= \overline{E}[Rew_B^{t-1}([S_i]_{i=1}^t) \mid S_1] = \\
&= \overline{E}[Rew_B^{t-1}([S_i]_{i=0}^{t-1}) \mid S_0].
\end{aligned}$$

Now, $\overline{E}[Rew_B^{t-1}([S_i]_{i=0}^{t-1}) \mid S_0 = s] = \overline{Exp} Rew_B^{t-1}(s)$ *for every* $s \in S_{=1}^B$ *by the inductive hypothesis. Plugging this back into the expression we obtained for* $\overline{E}[Rew_B^t([S_i]_{i=0}^t) \mid S_0 = s]$, *we have*

$$\begin{aligned}
&\overline{E}[Rew_B^t([S_i]_{i=0}^t) \mid S_0 = s] = \\
&\quad = rew(s) + I_{B^c}(s)\overline{E}[\overline{Exp} Rew_B^{t-1}(S_1) \mid S_0 = s] = \\
&\quad = rew(s) + I_{B^c}(s)[\overline{T}\,\overline{Exp} Rew_B^{t-1}](s) =: \overline{Exp} Rew_B^t(s),
\end{aligned}$$

for all $s \in S_{=1}^B$, *where the second step uses [97, Equation 6].*

By exploiting Theorem 12, we can now demonstrate the following result proving that the recursive schema above introduced converges to what expected.

Theorem 13 $\overline{E}[Rew_B \mid S_0]$ *restricted to* $S_{=1}^B$ *is a fixed point of the iterative scheme* (4.5).

Proof 7 *The proof follows the one of [97, Proposition 10] and considerations below. As we know from the proof of Theorem 4,* $\lim_{t\to+\infty} Rew_B^t = Rew_B$. *Hence, for every* $s \in \mathcal{S}_{=1}^B$, $\lim_{t\to+\infty} \overline{E}[Rew_B^t \mid S_0 = s] = \overline{E}[Rew_B \mid S_0 = s]$ *by [97, Proposition 7], since* $0 \leq Rew_B^0 \leq Rew_B^1 \leq \ldots \leq Rew_B$. *From Theorem 12 it also follows that* $\overline{Exp} Rew_B^t$ *are also non decreasing and non negative. Hence, by using as well the continuity of* $\overline{T}$ *with respect to non decreasing and non negative sequences, see [97, Lemma 1], we find that*

$$\begin{aligned}
&\overline{E}[Rew_B \mid S_0] \mid_{\mathcal{S}_{=1}^B} = \\
&\quad = \lim_{t\to+\infty} \overline{E}[Rew_B^t \mid S_0] \mid_{\mathcal{S}_{=1}^B} = \lim_{t\to+\infty} \overline{Exp} Rew_B^t = \\
&\quad = \lim_{t\to+\infty} (rew_{\mathcal{S}_{=1}^B} + I_{B^c}\overline{T}\,\overline{Exp} Rew_B^{t-1}) = \\
&\quad = rew_{\mathcal{S}_{=1}^B} + I_{B^c}\overline{T}\,\overline{E}[Rew_B \mid S_0] \mid_{\mathcal{S}_{=1}^B},
\end{aligned}$$

where in the last step we use again Theorem 12.

To conclude, let us focus on the imprecise counterpart of reward-bounded hitting probability h_B^r and its upper bound $\overline{h}_B^r$, which we defined as follows.

Definition 66 (Upper hitting probability) *Given a IMC $\mathcal{M}$, a set of states $B \subseteq \mathcal{S}$, a reward-threshold r, and an initial state $s \in \mathcal{S}$:*

$$h_B^r(s) := \max_{P_s^M \in K^{\mathcal{M}}} \sum_{\substack{\hat{\pi} \in \text{Paths}_{fin}(s) \,:\, \exists t \in \mathbb{N} \text{ s.t.} \\ \hat{\pi}(t) \in B \,\wedge\, \forall \tau < t\, \hat{\pi}(\tau) \notin B \,\wedge \\ rew(\hat{\pi}(0), \ldots, \hat{\pi}(t)) \leq r}} P_s^{M_{\mathsf{DTMC}}}(Cyl(\hat{\pi}))\,.$$

Similarly to the precise case (see Prop. 5), the values of $\overline{h}_B^{\rho}(s)$ for all $s \in \mathcal{S}$ and $\rho := 0, 1, \ldots, r$ provide a solution to the following system of equations:

$$\overline{h}_B^{\rho}(s) = \begin{cases} 1 & \text{if } s \in B \text{ and } rew(s) \leq r \\ 0 & \text{if } rew(s) > \rho \text{ or } s \notin \mathcal{S}_{>0} \\ \max\limits_{T(s,S') \in \mathcal{T}(s,S')} \sum\limits_{s' \in \mathcal{S}} T(s,s') \overline{h}_B^{\rho - rew(s)}(s') & \text{otherwise}\,. \end{cases} \tag{4.10}$$

To compute $\overline{h}_B^r$ we can therefore use a recursive schema analogous to that presented in Sec. 1.5.3. First, we define a matrix $\overline{\mathbf{h}}_B^{t,\rho_{0:r}}$ whose cells are the values of $\overline{h}_B^{t,\rho}(s)$ computed for each $s \in \mathcal{S}$ and for $\rho = 0, \ldots, r$.

For $t = 0$, we generate $\overline{\mathbf{h}}_B^{t,\rho_{0:r}}$ by computing the vectors $\overline{h}_B^{t=0,\rho}$ for $\rho = 0, \ldots, r$ as in Equation (1.17).

For increasing values of t, $\overline{\mathbf{h}}_B^{t,\rho_{0:r}}$ is generated by computing the vectors $\overline{h}_B^{t,\rho}$ for $\rho = 0, \ldots, r$ as follows:

$$\overline{h}_B^{t,\rho}(s) = \begin{cases} 1 & \text{if } s \in B \text{ and } rew(s) \leq \rho \\ 0 & \text{if } rew(s) > \rho \text{ or } s \notin \mathcal{S}_{>0} \\ \max\limits_{T(s,S') \in \mathcal{T}(s,S')} \sum\limits_{s' \in \mathcal{S}} T(s,s') h_B^{t-1,\rho - rew(s)}(s') & \text{otherwise}\,. \end{cases} \tag{4.11}$$

As in the precise case, the values of $h_B^{t-1,\rho-rew(s)}(s')$ for all $s' \in \mathcal{S}$ are provided by the matrix $\overline{\mathbf{h}}^{t-1,\rho_{0:r}}$ that we generate at time-step $t-1$. The recursion is based on iterated applications of the upper transition operator $\overline{\mathcal{T}}$, each one based on solving $|\mathcal{S}|$ linear programming tasks.

To compute $\overline{h}_B^r(s)$ for all $s \in \mathcal{S}$, we proceed as in the precise case, that is, we iterate $\overline{\mathbf{h}}_B^{t,\rho_{0:r}}$ over increasing values of t until convergence. Hence, for each $s \in \mathcal{S}$, $\overline{h}_B^r(s)$ is given by the s-cell of the r-vector of the matrix $\overline{\mathbf{h}}_B^{t^*,\rho_{0:r}}$, where t^* is the convergence time-step. The convergence of the procedure is granted by the following theorem:

Theorem 14 *Let $(\mathcal{M}, rew)$ be a IMRM, $B \subseteq [\mathcal{S}]$, and $\rho \in \mathbb{N}$, there is a $t^* \in \mathbb{N}$ such that for all $\tau \geq 0$:*

$$\overline{\mathbf{h}}_B^{t^*+\tau,\rho_{0:r}} = \overline{\mathbf{h}}_B^{t^*,\rho_{0:r}} . \tag{4.12}$$

Proof 8 *The proof is completely analogous to that of Theorem 6.*

The overall time complexity of the procedure is polynomial in $|\mathcal{S}|t^*$ for a reasoning analogous to that stated for Equation (4.4).

4.2.5 Imprecise Probabilistic Interpreted Systems

The second IMC extension we consider are *imprecise probabilistic interpreted systems* (IPISs) [158]. The latter are defined as multi-agent systems composed by agents whose stochastic behaviour is described in terms of IMCs. An IPIS under this interpretation is constructed as follows. For each agent $a \in \mathcal{A}$, let $\{K^a(S' \mid s)\}_{s\in\mathcal{S}}$ denote a family of CSs including, for each $s \in \mathcal{S}$, all the transition probability mass functions $P^a(S' \mid s)$ that are compatible with some agent's probabilistic beliefs. To obtain an IPIS, we replace all the transition matrices $T^a, a \in \mathcal{A}$ in a standard PIS with corresponding (row-stochastic) imprecise transition matrices $\mathcal{T}^a := \{K^a(S' \mid s)\}_{s\in\mathcal{S}}$, whose rows correspond to the transition CSs $K^a(S' \mid s)$ for all $s \in \mathcal{S}$. The overall stochastic behaviour of the entire multi-agent system is then described by a global *imprecise* transition matrix $\mathcal{T}_{\mathsf{IPIS}}$ which can be obtained following different approaches. The most conservative approach consists of defining $\mathcal{T}_{\mathsf{IPIS}}$ as a collection of $|\mathcal{S}|$ conditional CSs $K_{\mathsf{IPIS}}(S' \mid s)$, each one being defined as $\bigcup_{a\in\mathcal{A}} K^a(S' \mid s)$. While natural, this approach always implies an increase of the degree of imprecision, defined in terms of the size of the credal sets.

Another approach to obtain $\mathcal{T}_{\mathsf{IPIS}}$ consists of computing, for each transition $s, s' \in \mathcal{S} \times \mathcal{S}$, the credal version of the *logarithmic pooling* of the family of conditional CSs $\{K^a(S' \mid s) : a \in \mathcal{A}\}$. In general, this is defined as the element-wise application of the standard logarithmic pooling to the elements of the credal sets. This element-wise approach, however, might comport exponential complexity with respect to the number of agents in the model. A similar problem also occurs when considering alternative strategies, such as the one proposed in [7] within the framework of general credal networks. An efficient way to overcome the problem we adopt here consists in considering an outer

approximation of the lower and upper bounds of the credal logarithmic pooling achieved as follows:

$$\overline{\mathcal{T}}_{\mathsf{IPIS}}(s,s') := \frac{\prod_{a\in\mathcal{A}} \overline{\mathcal{T}}^{a}(s,s')}{\prod_{a\in\mathcal{A}} \overline{\mathcal{T}}^{a}(s,s') + \sum_{s''\neq s'} \prod_{a\in\mathcal{A}} \underline{\mathcal{T}}^{a}(s,s'')} \,. \tag{4.13}$$

The lower bound is analogously computed. The obtained global matrix $\mathcal{T}_{\mathsf{IPIS}}$ consists of an imprecise transition matrix $\mathcal{T}_{\mathsf{IPIS}}$ whose entries are intervals $(m,n) \subseteq [0,1]$ whose extremes are given by the lower $\underline{\mathcal{T}}_{\mathsf{IPIS}}(s,s')$ and the upper $\overline{\mathcal{T}}_{\mathsf{IPIS}}(s,s')$ bounds of the transition probabilities. As in the precise case, the global matrix describes the *embedded* IMC of the IPIS that is used to compute probabilistic inferences arising with the overall stochastic behaviour of the multi-agent system.

4.2.6 Imprecise Probabilistic Interpreted Reward Systems

The last class of IMC-based structures we consider are *imprecise probabilistic interpreted reward systems* (IPIRSs). The latter are defined as pairs $\langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle$ of a IPIS $\mathcal{M}_{\mathsf{IPIS}}$ and a reward function $rew : \mathcal{S} \mapsto \mathbb{N}$. As for IPISs, the global transition matrix $\mathcal{T}_{\mathsf{IPIRS}}$ of an IPIRS is obtained by combining the credal transition matrices $\mathcal{T}^{a}$ of the various agents $i \in \mathcal{A}$.

Notice that all the various imprecise Markov models above introduced can be seen as components of an IPIRS. Specifically, an IPIRS $(\mathcal{M}_{\mathsf{IPIS}}, rew)$ includes an IPIS $\mathcal{M}_{\mathsf{IPIS}}$ that is composed by several IMCs $\mathcal{M}$, one for each agent of the system. On the other hand, an IPIRS $(\mathcal{M}_{\mathsf{IPIS}}, rew)$ includes several IMRMs, one for each agent in the system, composed by an IMC $\mathcal{M}$ and the reward function rew. This work focuses on only IPIRS and their properties. Nevertheless, the various results obtained for IPIRSs can be easily transferred to IMCs, IMRMs, and IPISs.

4.3 Epistemic Imprecise PRCTL

This section presents *Epistemic Imprecise Probabilistic Reward Computation Tree Logic* (EIPRCTL), an epistemic and imprecise-probabilistic extension of the PCTL introduced in [78] suitable to specify epistemic, probabilistic, and reward properties of non-stationary (or not fully specifiable) stochastic multi-agent systems. The language is targeted on IPIRSs but can be also applied to the other kinds of imprecise Markov models previously introduced. Notably, properties of IMCs, IMRMs, and IPISs can be specified via specific languages, such as IPCTL [163], IPRCTL [157], and EIPCTL [158], which can be obtained as fragments of EIPRCTL.

4.3.1 EIPRCTL Syntax

The EIPRCTL syntax is recursively defined as follows:

$$\phi := \begin{cases} \top \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \underline{P}_{\nabla b}\psi \mid \overline{P}_{\nabla b}\psi \mid P_J\psi \\ \underline{E}_{\nabla r}\phi \mid \overline{E}_{\nabla r}\phi \mid E_J\phi \mid K^a\phi \mid E^\Gamma\phi \mid C^\Gamma\phi \mid D^\Gamma\phi \end{cases},$$

$$\psi := \bigcirc\phi \mid \phi_1 \bigcup \phi_2 \mid \phi_1 \bigcup^{\leq t} \phi_2 \mid \phi_1 \bigcup_{\leq r} \phi_2\,,$$

$$\epsilon := B^a_{\nabla \underline{b}}\phi \mid B^a_{\nabla \overline{b}}\phi\,,$$

where $p \in AP$, $b \in [0,1]$, $J \subseteq [0,1]$, $a \in \mathcal{A}$, $\Gamma \subseteq \mathcal{A}$, and ∇ is a short notation for $\{<, \leq, =, \geq, >\}$.

The language is composed by ϕ, ψ and ϵ-formulae. The former extend classical propositional logic with usual operators for single-agent knowledge K^a, *common knowledge* C^Γ, and *distributed knowledge* D^Γ, and with the following probabilistic modalities:

- $\underline{P}_{\nabla b}\psi$: *the lower probability of reaching a path that satisfies* ψ *is* ∇b;
- $\overline{P}_{\nabla b}\psi$: *the upper probability of reaching a path that satisfies* ψ *is* ∇b;
- $P_J\psi$: *the probability of reaching a path that satisfies* ψ *belongs to the closed interval* $J \subseteq [0,1]$;
- $\underline{E}_{\nabla r}\phi$: *the lower bound of the expected cumulative reward earned by the system until reaching a state that satisfies* ϕ *is* ∇r;
- $\overline{E}_{\nabla r}\phi$: *the upper bound of the expected cumulative reward earned by the system until reaching a state that satisfies* ϕ *is* ∇r;
- $E_J\phi$: *the expected cumulative reward earned by the system until reaching a state that satisfies* ϕ *belongs to the closed interval* $J \subseteq [0,1]$;

The ψ-formulae are standard CTL path-formulae [10, p.313] used to represent properties of paths:

- $\bigcirc\phi$: *in the next state of the path* ϕ *holds*;
- $\phi_1 \bigcup \phi_2$ ϕ_1: ϕ_1 *holds along the path until* ϕ_2 *holds*;
- $\phi_1 \bigcup^{\leq t} \phi_2$: *there exists a time-step* $\tau \leq t$ *such that* ϕ_2 *holds in the* τ*-step of the path and* ϕ_1 *holds in all the previous time-steps*;

- $\phi_1 \bigcup_{\leq r} \phi_2$: *$\phi_1$ holds in all states of the path until a cumulative reward lower then or equals to r is earned then ϕ_2 holds.*

Finally, ϵ-formulae include the two following *weighted-belief* modalities[5]:

- $B^a_{\nabla \underline{b}}\phi$: *the lower bound of the agent a degree of belief that ϕ will be reached eventually in the future is ∇b;*
- $B^a_{\nabla \overline{b}}\phi$: *the upper bound of the agent a degree of belief that ϕ will be reached eventually in the future is ∇b.*

4.3.2 EIPRCTL Semantics

Let us introduce a proper semantics for EIPRCTL formulae based on IPIRSs. This can be seen as a generalisation of the semantics proposed in [163] for IMCs and those proposed in [175] for standard (precise) PISs.

Semantics of state-formulae

We start by presenting satisfiability conditions for Boolean, probabilistic, and epistemic ϕ-formulae separately.

Definition 67 (Semantics of Boolean formulae) *Given an IPIRS $\langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle$ and a state $s \in \mathcal{S}$, the following conditions hold:*

$$\begin{array}{ll} \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models p & \text{iff } p \in l(s)\,, \\ \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models \neg\phi & \text{iff } \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \not\models \phi\,, \\ \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models \phi_1 \wedge \phi_2 & \text{iff } \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models \phi_1 \text{ and } \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models \phi_2\,. \end{array}$$

Definition 68 (Semantics of probabilistic formulae) *Given an IPIRS $\langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle$ and a state $s \in \mathcal{S}$, the following conditions hold:*

$$\begin{array}{ll} \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models \underline{P}_{\nabla b}\psi & \text{iff } \underline{P}_{\mathsf{IPIRS}}(s \models \psi)\nabla b\,, \\ \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models \overline{P}_{\nabla b}\psi & \text{iff } \overline{P}_{\mathsf{IPIRS}}(s \models \psi)\nabla b\,, \\ \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models P_J\psi & \text{iff } \begin{cases} \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models \underline{P}_{\geq \min J}\psi \text{ and} \\ \langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models \overline{P}_{\leq \max J}\psi\,. \end{cases} \end{array}$$

For the probabilistic formulae, the satisfiability conditions refer to the lower and upper bounds of $P_{\mathsf{IPIRS}}(s \models \psi)$, where $P_{\mathsf{IPIRS}}(s \models \psi)$ is the probability that a path $\pi \models \psi$ belongs

[5] In the rest of this work, we refer to such doxastic formulae as *imprecise probabilistic beliefs*.

to the set of paths originating in s conditional to $S_0 = s$.[6] Similarly to standard PCTL [10], the computation of the lower and upper bounds of $P_{\mathsf{IPIRS}}(s \models \psi)$ varies depending on ψ. We analyse further this point in Section 4.4 dedicated to model checking procedures.

Definition 69 (Semantics of expected reward formulae) *Let $Sat(\phi)$ be the set of all states that satisfy ϕ. Given an IPISR $(\mathcal{M}_{\mathsf{IPIS}}, rew)$ and state $s \in \mathcal{S}$, the following condition holds:*

$$\begin{array}{ll}
\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, s \models \overline{E}_{\nabla r}\phi & \text{iff } \overline{Exp}Rew_{Sat(\phi)}(s)\nabla r\,, \\
\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, s \models \underline{E}_{\nabla r}\phi & \text{iff } \underline{Exp}Rew_{Sat(\phi)}(s)\nabla r\,, \\
\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, s \models E_J\phi & \text{iff } \begin{cases} \langle \overline{\mathcal{M}}_{\mathsf{IPIS}}, rew\rangle, s \models \underline{E}_{\geq \min J}\phi \text{ and} \\ \langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, s \models \overline{E}_{\leq \max J}\phi\,. \end{cases}
\end{array}$$

.

Definition 70 (Semantics of epistemic formulae) *Given an IPIRS $(\mathcal{M}_{\mathsf{IPIS}}, rew)$, an agent $i \in \mathcal{A}$ or a group of agents $\Gamma \subseteq \mathcal{A}$, and a state $s \in S$, the following conditions hold:*

$$\begin{array}{ll}
\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, s \models K^a\phi & \text{iff } \forall s', s \sim^a s' : s' \models \phi\,, \\
\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, s \models E^\Gamma\phi & \text{iff } \forall s', s \sim_E^\Gamma s' : s' \models \phi\,, \\
\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, s \models C^\Gamma\phi & \text{iff } \forall s', s \sim_C^\Gamma s' : s' \models \phi\,, \\
\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, s \models D^\Gamma\phi & \text{iff } \forall s', s \sim_D^\Gamma s' : s' \models \phi\,.
\end{array}$$

Semantics of path-formulae

Definition 71 (Semantics of ψ-formulae) *Given an IPIRS $(\mathcal{M}_{\mathsf{IPIS}}, rew)$ and a path π, the following conditions hold:*

$$\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi \models \bigcirc\phi \quad \text{iff } \langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi(1) \models \phi\,,$$

$$\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi \models \phi_1 \bigcup^{\leq t} \phi_2 \quad \text{iff } \exists \tau \leq t : \begin{cases} \langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi(\tau) \models \phi_2 \text{ and} \\ \forall \tau' < \tau \ : \ \langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi(\tau) \models \phi_1\,, \end{cases}$$

$$\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi \models \phi_1 \bigcup \phi_2 \quad \text{iff } \exists t \geq 0 : \begin{cases} \langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi(t) \models \phi_2 \text{ and} \\ \forall \tau : 0 \leq \tau < t \ \langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi(\tau) \models \phi_1\,, \end{cases}$$

$$\langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi \models \phi_1 \bigcup_{\leq r} \phi_2 \quad \text{iff } \exists t \in \mathbb{N} : \begin{cases} \langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi(t) \vDash \phi_2 \text{ and} \\ \forall \tau < t \ : \ \langle \mathcal{M}_{\mathsf{IPIS}}, rew\rangle, \pi(\tau) \vDash \phi_1 \\ \text{and } Rew(\pi, t) \leq r\,. \end{cases}$$

[6] Here the subscript IPIRS denotes the fact that this probability is computed through the global transition matrix $\mathcal{T}_{\mathsf{IPIRS}}$ describing the stochastic behavior of the whole multi-agent system.

Semantics of weighted-belief formulae

The ϵ-formulae are doxastic formulae that specify the imprecise *degree of belief* of a specific agent concerning the overall behaviour of the multi-agent system. In this regard, they qualify as the imprecise "analogous" of the COGWED weighted belief formulae reported in Chapter 1, Section 2.11. In COGWED, the degree of belief for a given formula ϕ and a set of agents Γ is measured as the ratio between the respective *steady-state* probabilities of the events $Sat(\phi) \cap eq^{\sim^{\Gamma}_{D}}(s)$ (i.e., the set of states that satisfy ϕ and at the same time belong to the distributed-knowledge epistemic equivalence class of s) and $eq^{\sim^{\Gamma}_{D}}(s)$. Here, we adopt an analogous definition but considering imprecise upper (lower) steady-state probabilities instead of their standard precise counterparts.

Analogously to standard Markov chains, the upper (lower) steady state probability distribution of an IMC $\mathcal{M}$ relative to a given event $B \subseteq \mathcal{S}$ can be obtained as $\overline{\iota}^{\mathcal{M}} \cdot \overline{h}_B$. For a given state $s \in \mathcal{S}$, the upper (lower) bound of the imprecise degree of belief relative to a group of agent Γ and a formula ϕ is accordingly defined as the ratio between $\overline{\iota}^{\mathcal{M}}(s) \cdot \overline{h}_{Sat(\phi) \cap eq^{\sim^{\Gamma}_{D}}(s)}(s)$ and $\overline{\iota}^{\mathcal{M}}(s) \cdot \overline{h}_B(s)$. From this definition, we then obtain the following satisfiability conditions for ϵ-formulae.

Definition 72 (Satisfiability of ϵ-formulae) *Given an IPIRS $(\mathcal{M}_{\mathsf{IPIS}}, rew)$, and a state $s \in \mathcal{S}$,*

$$\langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models B^{a}_{\nabla \underline{b}} \phi \quad \text{iff} \quad \frac{\underline{\iota}^{\mathcal{M}}(s) \cdot \underline{h}_{Sat(\phi) \cap eq^{\sim^{\Gamma}_{D}}(s)}(s)}{\underline{\iota}^{\mathcal{M}}(s) \cdot \underline{h}_B(s)},$$

$$\langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models B^{a}_{\nabla \underline{b}} \phi \quad \text{iff} \quad \frac{\overline{\iota}^{\mathcal{M}}(s) \cdot \overline{h}_{Sat(\phi) \cap eq^{\sim^{\Gamma}_{D}}(s)}(s)}{\overline{\iota}^{\mathcal{M}}(s) \cdot \overline{h}_B(s)}$$

4.4 Model Checking

In the present section, we explain how to check systems modelled by IPISRs against properties specified in the EIPRCTL language. The procedure we present is obtained by extending the *parsing-tree* algorithm for CTL presented in Chapter 2, Section 2.6.2. We start by briefly recalling the structure and functioning of the parsing-tree. Then, we extend the algorithm introducing a series of new sub-routines to solve specific model checking tasks related to the different kinds of EIPRCTL formulae.

4.4.1 Parsing Tree

Let Λ be a short notation for either a ϕ-formula or an ϵ-formula. Given an IPISR $\langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle$, a state $s \in \mathcal{S}$, and a formula Λ, we define an algorithm that verifies whether

$\langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle, s \models \Lambda$. The algorithm works as follows:

1. Generate the *parse tree* of Λ, recursively decomposing Λ in its sub-formulae λ until only atoms remain.
2. Traverse the parse tree of Λ visiting all the sub-formulae λ, starting from the leaves and working backwards to the roots,
3. At each sub-formula λ, calculate the set of states that satisfy λ, denoted $Sat(\lambda)$, by checking whether $s \models \lambda$ for all $s \in \mathcal{S}$,
4. Calculate $Sat(\Lambda)$ by composition of the various $Sat(\lambda)$,
5. Check whether $s \in Sat(\Lambda)$.

The algorithm includes a specific sub-routine to compute $Sat(\Lambda)$ for each specific kind of EIPRCTL formula λ, as following detailed.

Boolean formulae.

For Boolean formulae, $Sat(\lambda)$ is computed as follows:

$$
\begin{aligned}
&Sat(\top) := \mathcal{S}\,,\\
&Sat(p) := \{s \in \mathcal{S} : p \in l^i(s)\}\,,\\
&Sat(\phi_1 \wedge \phi_2) := Sat(\phi_1) \cap Sat(\phi_2)\,,\\
&Sat(\neg\phi) := \mathcal{S} \setminus Sat(\phi)\,.
\end{aligned}
$$

Probabilistic formulae.

For formulae of the kind $\underline{P}_{\nabla b}\psi$, $\overline{P}_{\nabla b}\psi$ and $P_J\psi$, the set $Sat(\lambda)$ is obtained by computing the lower and upper bounds of $P_{\mathsf{IPIS}}(s \models \psi)$ for each $s \in \mathcal{S}$ and then check whether they satisfy the specified threshold ∇b. The specific procedure to compute the lower and upper bounds of $P_{\mathsf{IPIS}}(s \models \psi)$ varies depending on the specification of ψ.

Next If $\psi := \bigcirc\phi$, then $P_{\mathsf{IPIRS}}(s \models \psi)$ corresponds to $P_s(s_1 \in Sat(\phi))$ and its upper (lower) bound can be computed as in Equation (4.2).

Time-bounded Until If $\psi = \phi_1 \bigcup^{\leq t} \phi_2$, then $P_{\mathsf{IPIRS}}(s \models \psi)$ corresponds to the probability of hitting $Sat(\phi_2)$ within a finite number of time-steps t conditional on $S_0 = s$ and with the additional condition that all the states visited before reaching $Sat(\phi_2)$ are in $Sat(\phi_1)$. For each $s \in \mathcal{S}$, we denote such probability by $h^t_{Sat(\phi_2)|Sat(\phi_1)}(s)$. A recursive schema

analogous to that in Equation (4.4) can be formulated to compute $\overline{h}^{t}_{Sat(\phi_2)|Sat(\phi_1)}$. Let $\mathbb{I}_{Sat(\phi_1)\setminus Sat(\phi_2)}$ denote the indicator vector whose values are one for all $s \in Sat(\phi_1)\setminus Sat(\phi_2)$ and 0 otherwise. A slightly modified version of the algorithm in (4.4) for computing $\overline{h}^{t}_{Sat(\phi_2)|Sat(\phi_1)}$ by recursion over increasing values of t is achieved as follows:

$$\overline{h}^{t}_{Sat(\phi_2)|Sat(\phi_1)} := \mathbb{I}_{Sat(\phi_2)} + \mathbb{I}_{Sat(\phi_1)\setminus Sat(\phi_2)}(\overline{\mathcal{T}}_{\mathsf{IPIS}}\, \overline{h}^{t-1}_{Sat(\phi_2)|Sat(\phi_1)})\,. \tag{4.14}$$

As in Equation (4.4), the initialisation is given by the indicator function of $Sat(\phi_2)$ while the recursive steps consist of iterated applications of the upper transition operator to the hitting vector computed at the precedent time-step $t-1$. The only relevant difference with the analogous scheme presented in Sec. 4.2.2 consists of the indicator vector $\mathbb{I}_{Sat(\phi_1)\setminus Sat(\phi_2)}$ that replaces the indicator vector $\mathbb{I}_{B^c}$ of the complement of the hitting event B. In the general scheme, $\mathbb{I}_{B^c}$ limits the iteration considering only paths that have not already visited an $s \in B$. Here, by using $\mathbb{I}_{Sat(\phi_1)\setminus Sat(\phi_2)}$, we limit the iteration to only those paths whose actual and previous states are all in $Sat(\phi_1)$ and that have not already reached a state $s \in Sat(\phi_2)$. Notice that Equation (4.14) is the imprecise analogous of the system of linear equations used to compute in $P(s \models \phi_1 \bigcup \phi_2)$ in the precise case, as reported in [10, Sec. 10.2.1]. Finally, as for Equation (4.4), the computation of Equation (4.14) is based on solving $|\mathcal{S}|t$ linear programming tasks and its time complexity is, thus, polynomial in $|\mathcal{S}|$ t.

Until If $\psi := \phi_1 \bigcup \phi_2$, then $P_{\mathsf{IPIRS}}(s \models \psi)$ corresponds to the probability of hitting $Sat(\phi_2)$ conditional on $S_0 = s$ and with the additional requirement that all the states visited before reaching $Sat(\phi_2)$ are in $Sat(\phi_1)$. To compute the lower and upper bounds of this probability, we simply iterate the schema in Equation (4.14) over increasing values of t until convergence.

Reward-bounded Until If $\psi := \phi_1 \bigcup_{\leq r} \phi_2$, then $P_{\mathsf{IPIRS}}(s \models \phi_1 \bigcup_{\leq r} \phi_2)$ corresponds to the reward-bounded hitting probability of $Sat(\phi_2)$ with the additional condition that all the states visited before reaching $Sat(\phi_2)$ are in $Sat(\phi_1)$. We denote this probability by $h^{r}_{Sat(\phi_2)|Sat(\phi_1)}$. To compute the upper (lower) bound of the latter, we involve a sightly modified version of the procedure introduced in Equation (4.10) and (4.11) .

Let $\overline{\mathbf{h}}^{t,\rho_{0:r}}_{Sat(\phi_2)|Sat(\phi_1)}$ be a matrix whose cells are the values of $\overline{h}^{t,\rho}_{Sat(\phi_2)|Sat(\phi_1)}(s)$ computed for each $s \in \mathcal{S}$ and for $\rho : 0, 1, \ldots, r$.

For $t = 0$, we generate $\overline{\mathbf{h}}^{t,\rho_{0:r}}_{Sat(\phi_2)|Sat(\phi_1)}$ by computing the vectors $\overline{h}^{t=0,\rho}_{Sat(\phi_2)|Sat(\phi_1)}$ for $\rho = 0, 1, \ldots, \rho.$ as in Equation (1.17).

For $t > 0$, we generate $\overline{\mathbf{h}}^{t,\rho_{0:r}}_{Sat(\phi_2)|Sat(\phi_1)}$ by computing the various vectors $\overline{h}^{t,\rho}_{B}$ for $\rho =$

$0, 1, \ldots, r$ as follows:

$$\overline{h}^{t,\rho}_{Sat(\phi_2)|Sat(\phi_1)}(s) = \begin{cases} 1 \text{ if } s \in Sat(\phi_2) \text{ and } rew(s) \leq \rho\,, \\ 0 \text{ if } \underline{h}(s) = 0 \text{ or } s \notin Sat(\phi_1) \setminus Sat(\phi_2) \text{ or } rew(s) > \rho\,, \\ \overline{\mathcal{T}} h_B^{t-1,\rho-rew(s)}(s) \text{ otherwise}\,. \end{cases} \quad (4.15)$$

The schema is analogous to that in Equation (4.11). The only relevant difference is the additional clause prescribing that $\overline{h}^{t,\rho}_B(s) = 0$ also for all $s \in Sat(\phi_1) \setminus Sat(\phi_2)$, whereas in Equation (4.11) $\overline{h}^{t,\rho}_B(s) = 0$ only for $s \in \mathcal{S}$ such that either $\underline{h}_B(s') = 0$ or $rew(s) > r$. The additional clause blocks the recursion for the successors of the initial state that do not belong to $Sat(\phi_1)$. Indeed, if a certain successor $s' \notin Sat(\phi_1) \setminus Sat(\phi_2)$ is reached at a certain time-step τ of the iteration, then $h^r_{Sat(\phi_2)|Sat(\phi_1)}(s')$ takes value zero and the recursion from that state is stopped. In such a way, it is possible to account for the additional requirement that all states visited before reaching the hitting event $Sat(\phi_2)$ are in $Sat(\phi_1)$. Notice that the slightly modification does not alter the general results about $\overline{h}^r_B$ reported above. In particular, the time complexity of the procedure remains polynomial in $|\mathcal{S}|t^*$ (with t^* denoting the convergence time-steps) as it practically depends on the iterative step $\overline{\mathcal{T}} h_B^{t-1,\rho-rew(s)}(s)$, which is the same both in Equation (4.11) and (4.15).

Expected reward formulae.

For formulae of the kind $\underline{E}_{\nabla r}\phi \mid \overline{E}_{\nabla r}\phi \mid E_J\phi$, the procedure to determine $Sat(\lambda)$ is based on computing the lower or the upper bounds of $ExpRew_{Sat(\lambda)}(s)$, with $Sat(\lambda) \subseteq \mathcal{S}$ following the procedure in Equation (4.5).

Epistemic Formulae

For formulae of the kind $K^i\phi \mid C^\Gamma\phi \mid D^\Gamma\phi$, the sub-routine to compute $Sat(\lambda)$ is reported in Fig. 8.

Algorithm 4: $Sat(\kappa\phi)$

Input: A IPIRS $\mathcal{M}_{\mathsf{IPIRS}}$ and a formula $\kappa\phi$
Output: $Sat(\kappa\phi)$

```
Sat(κφ) ← {}
Compute Sat(φ) := {s ∈ S | M_IPIRS, s ⊨ φ}
foreach eq^{~^κ} ∈ Eq^{~^κ} do
    if eq^{~^κ} ⊆ Sat(φ) then
        Sat(κφ) ← {eq^{~^κ}}
    end
end
return Sat(κφ)
```

Notice that this sub-routine is analogous to the one introduced in Section 2.9.2 for CTLK epistemic formulae. It works as follows:

1. It takes in input an IPIRS M_{IPIRS} and a formula $\kappa\phi$;

2. It computes $Sat(\phi)$ by recursively calling the respective subroutine;

3. For each $eq^{\sim^\kappa} \in Eq^{\sim^\kappa}$, it checks whether $eq^{\sim^\kappa} \subseteq Sat(\phi)$. If this is the case, then the algorithm adds the whole equivalence class $eq^{\sim^\kappa}$ to $Sat(\kappa\phi)$.

Imprecise weighted-belief formulae For ϵ-formulae, the procedure to compute $Sat(\lambda)$ is analogous to the one introduced in Section 2.11.2 for COGWED weighted-belief formulae within the PIS-based semantic framework. The only relevant different is the replacement of the initial probability vector $\iota^{\mathcal{M}_{\mathsf{PIS}}}$ and the hitting vector $h^{\mathcal{M}_{\mathsf{PIS}}}$ with their imprecise analogous $\overline{\iota}^{\mathcal{M}_{\mathsf{IPIS}}}$ and $\overline{h}^{\mathcal{M}_{\mathsf{IPIS}}}$ (or the analogous for lower probabilities). The algorithm is reported in Figure 12.

Algorithm 12: $Sat(B^{\Gamma}_{\nabla b}\phi)$

Input: A PIS M_{IS} and a COGWED weighted-doxastic formula $B_{\nabla b}\phi$
Output: $Sat(B^{\Gamma}_{\nabla b}\phi)$

$Sat(B^{\Gamma}_{\nabla b}\phi) \leftarrow \{\}$
Compute $Sat(\phi) := \{s \in \mathcal{S} \mid M_{\mathsf{IS}}, s \models \phi\}$
Compute the partition $Eq^{\sim^{\Gamma}_{D}}(s)$
foreach $eq^{\sim^{\Gamma}_{D}}(s) \in Eq^{\sim^{\Gamma}_{D}}(s)$ **do**
  Compute $\overline{h}^{M_{\mathsf{PIS}}}_{Sat(\phi)\cap eq^{\sim^{\Gamma}_{D}}(s)}(s)$ through the schema in Equation (1.8)
  Compute $\overline{h}^{M_{\mathsf{PIS}}}_{eq^{\sim^{\Gamma}_{D}}(s)}(s)$ through the schema in Equation (1.8)
  if $\frac{\bar{\imath}^{M_{\mathsf{PIS}}}(s)\overline{h}^{M_{\mathsf{PIS}}}_{Sat(\phi)\cap eq^{\sim^{\Gamma}_{D}}(s)}(s)}{\bar{\imath}^{M_{\mathsf{PIS}}}(s)\overline{h}^{M_{\mathsf{PIS}}}_{eq^{\sim^{\Gamma}_{D}}(s)}(s)} \nabla b$ **then**
    $Sat(B^{\Gamma}_{\nabla b}\phi) \leftarrow \{eq^{\sim^{\Gamma}_{D}}(s)\}$
  end
end
return $Sat(B^{\Gamma}_{\nabla b}\phi)$

Figure 4.1

4.5 A Case Study on Healthcare Budgeting

We present a first validation of EIPRCTL based on a slightly modified version of the MRM originally proposed in [117]. MRMs are used in that work to estimate the recovery costs for patients in geriatric departments. Two kinds of recovery are considered: short-term recoveries for acute cares have a daily cost estimated as £100, while long-term recoveries cost £50 per day. From a cumulative perspective, long-term recoveries are more expensive, since those patients typically remain in the hospitals for longer periods.

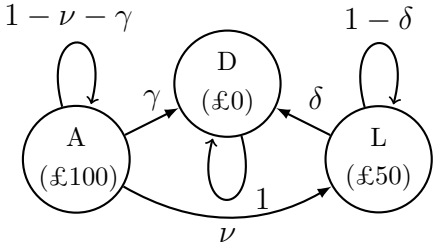


Figure 4.2: Transitions in a three-state MRM.

The evolution across time of health conditions of a patient can be described through a MC with three states: acute care A, long-term care L, and discharge or death D. Transitions from L to A are considered impossible, while D acts as an absorbing state. A parametrized version of the transition matrix for this model is in Figure 4.2. The parameters have the following interpretation: the *conversion rate* ν corresponds to the probability of passing from a short-term to a long-term recovery, while the *dismissing rates* γ and δ correspond to the probability of being discharged/die, respectively, in a short and long-term recovery. Rates γ, ν and δ vary depending on the patient and disease. An assessment of these parameters for three different departments is in Table 4.1.

Rate (%)	Department 1	Department 2	Department 3
γ	1.750	3.540	2.810
ν	0.031	0.187	0.149
δ	0.120	0.130	0.180

Table 4.1: Conversion and dismissing rates.

The reward rew associated with each state represents the daily cost per patient. In a scale where one is assumed to correspond to one pound, the daily costs per patient are described by a function such that $rew(A) = 100$, $rew(L) = 50$, and $rew(D) = 0$. When a patient is dismissed or death she no longer has a cost for the hospital. Following these specifications, it is possible to construct a (precise) MRM $\langle M, rew \rangle$ able to predict the expected cumulative cost incurred by the hospital for each patient up to the time of the patient's discharge or death. Suppose that the total amount of financial resources per patient available to the hospital is $\rho := \$40,000$. We are interested in verifying whether the expected cumulative cost per patient is sustainable, i.e., it does not exceed the amount of resources available. This corresponds to check whether:

$$ExpRew_D(s) \leq \rho\,,$$

for both $s = A$, i.e., for a patient initially recovered in *acute care*, and $s = L$, i.e., for a patient initially recovered in *long-term care*.

The MRM in [117] presents an important limitation, that is, it considers precise values for the transition rates ν, γ and δ, which is tantamount to assuming that the probability of a patient to change her health-condition is always the same independently from time. This assumption is clearly problematic. For example, it is obvious that the probability to die for patients in long-term cares increases with time. Imprecise probabilistic models allow us to overcome this limitation by considering, instead of precise values, probability intervals within which the transition rates can fluctuate over time.

In practice, we define an imprecise transition matrix for each department obtained by a perturbation of the values of each column of Table 4.1. As a perturbation method for converting a probability mass function into a credal set, we simply adopt a *linear-vacuous* contamination [173]. The methods works as follows. If $P(S)$ is a PMF over S, the CS $K(S)$ (see Section 4.2.1) becomes the CS that includes all the PMFs obtained as a mixture $(1-\alpha)P(S) + \alpha P'(S)$, where $P'(S)$ is any probability mass function and the parameter $\alpha \in [0,1]$ defines the level of imprecision in the CS (e.g., $K(S) = \{P(S)\}$ for $\alpha = 0$, while for $\alpha = 1$ we get the vacuous CS). In our example, we assume a perturbation level $\alpha = 0.03$. Hence, by applying the above contamination model to each row of the three precise transition matrices described in Table 4.1, we obtain three imprecise transition matrices $\mathcal{T}^a, a \in \{1,2,3\}$ (where a is the number of the department to which the matrix refers).

Given that the costs associated with each state in the model remain the same as above, we have now obtained three different IMRMs, each describing the scenario related to one of the three departments. We are still interested in checking whether, given a patient, the expected cumulative cost incurred until the patient is dismissed or dies does not overcome the available resources. However, we neither know from which one of the three departments the patient comes nor whether it is recovered in acute (A) or long-term (L) care. We can model this scenario as follows.

First, instead of considering a specific transition matrix $\mathcal{T}^a, a \in \{1,2,3\}$, we consider an aggregated model where the global imprecise transition matrix is a pooling of the imprecise transition matrices of each department.

Such a global imprecise transition matrix $\mathcal{T}_{\text{IPIS}}$ can be obtained by the logarithmic pooling as in Eq.(4.13). As an alternative, more cautious, estimate, we also consider instead a *conservative* pooling consisting in taking as aggregated model the union of the probability intervals of the imprecise transition matrices of the different departments.

Finally, the fact that, independently from which is the department the patient comes from, we cannot know whether the patient is recovered in acute or long-term cares corresponds to assume that $A \sim^a L, \forall a \in \{1,2,3\}$, i.e., states labelled by A and L are indistinguishable. We have now obtained a description of the considered scenario in terms of an IPISR $\langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle$.

Checking whether the upper maximum expected cumulative cost incurred by the hospital until a patient is dismissed or dies is sustainable corresponds therefore to verify whether or not the formula $\overline{E}_{\leq \rho} D$ holds in the model $\langle \mathcal{M}_{\mathsf{IPIS}}, rew \rangle$ for each state s in the equivalence class $\{A, L\}$. To check this formula, we apply the procedure discussed in Sec. 4.4. The algorithms described in Sec. 4.2 are finally used to compute the upper bounds of $ExpRew_D(s)$ for both $s = A$ and $s = L$. The most cautious bounds returned by the

conservative pooling are:

$$\overline{Exp}Rew_D(A) = \$29'561\,, \tag{4.16}$$
$$\overline{Exp}Rew_D(L) = \$42'343\,. \tag{4.17}$$

As expected, the cumulative costs for patients initially admitted in acute-care are lower and not exceeding the resources available to the hospital. The same does not happen for patients initially admitted in long-term care.

The management of the hospital might consequently need to check how likely is the fact that the cumulative costs for an hospitalised patient are exceeding the available resources. We do that by checking that the probability of a patient to be dismissed/died before the cumulative cost overcomes the available resources is sufficiently high, e.g., greater than or equal to a threshold $\pi := 0.95$. This corresponds to check whether the formula $\underline{P}_{\geq\pi}\top\bigcup_{\leq\rho} D$ holds in the model $\langle\mathcal{M}_{\mathsf{IPIS}}, rew\rangle$ and for each state s in the equivalence class $\{A, L\}$. Remarkably, by means of the algorithms described in Section 4.2 we obtain that the formula is satisfied for both initial states. The resources overrun is therefore a relatively unlikely event for the hospital.

4.6 Conclusions

An intrinsic limitation of probabilistic model checking is related to its fundamental reliance on the use of standard Markov models, which can notoriously model only stationary agents whose transition probabilities are all specified by known numerical values. To overcome this limitation, we have presented a novel framework based on the theory of imprecise probabilities and the related imprecise Markov models. More specifically, we have explained how the use of imprecise Markov models allows us to apply probabilistic model checking methods to both non-stationary agents and agents whose transition probabilities are not fully known without comporting computational complexity issues. The key point is that both probabilistic and reward inferences in imprecise Markov models can be computed by iteratively solving linear programming tasks. This allows us to solve relevant model checking tasks without increasing the computational complexity of the relative procedures, which always remains polynomial in the number of states of the models. The chapter focuses specifically on stochastic multi agent systems, but the framework it introduces is useful also for single-agent models. The main limitation is that, so far, we considered only discrete-time models. Recent developments [96] in the study of imprecise *continuous-time* Markov chains (CTMC) strongly suggest that an analogous framework can be introduced for continuous-time models, which are of fundamental relevance for applications in fields like computational and systems biology [22, 14]. In the model checking community, some

works concerning non-stationarity issues in continuous-time Markov models have been recently proposed. In [24], for example, non-stationary agents are modelled via *uncertain* CTMCs, which are CTMCs whose transition probabilities vary non-deterministically in time within bounded continuous intervals. Although uncertain and imprecise CTMCs are similar, they are not equivalent formalisms. Notably, while an uncertain CTMC can be regarded as the largest family of precise CTMCs compatible with the bounds of the intervals, an imprecise CTMC is the largest family of all, potentially non-Markov, non-homogeneous, processes compatible with given constraints. In other words, imprecise CTMCs are more expressive and potentially useful for a wider range of applications. So far, however, there are no works specifically concerning model checking with imprecise CTMCs.

To conclude, another important development to consider might concern the development of an imprecise-probabilistic framework for Markov decision-processes, notably for the relevance of the latter and their natural connection with the field of Reinforcement Learning [154] that we will involve in the last chapter of this thesis.

Chapter 5

Markov Models Semantics and Logical Omniscience

Abstract

This chapter explores an innovative application of Markov model semantics to solve the well-known and widely debated problem of logical omniscience. The chapter begins with an overview of the most relevant proposed solutions to this problem. It argues that almost all of these solutions are unsatisfactory because they fail to represent real-world agents as both logically competent and non-omniscient in virtue of their limited inferential abilities and resources. Subsequently, the chapter presents a novel solution based on explicitly representing real-world agents' knowledge as encompassing all and only the information they can infer using their limited inferential resources and abilities. To capture this new concept of explicit knowledge, the chapter introduces a semantic framework based on a new typology of Markov models called *Markov deduction structures*. The latter model deductive inference as a state-space exploration task, which is represented as a Markov decision process whose states are finite sets of Boolean formulae, and the actions correspond to logical rules. Reinforcement learning algorithms are then applied to solve this decision process and determine if an agent can effectively infer certain information without exceeding its available resources. The second part of the chapter introduces the *Markov Deduction Structures Logic* (MDSL), a new epistemic language defined based on the aforementioned semantics of Markov deduction structures, and its a dynamic extension (MDSDL). The latter includes specific operators for real-world agents' relevant actions such as learning a new rule, sharing resources, and exchanging information.

5.1 Introduction

The problem of *logical omniscience* [150] arguably represents one of the historically most famous and discussed topics in the philosophy of epistemic logic. This problem arises from the fact that knowledge and belief in the standard semantic framework of epistemic logic

are represented as closed under logical consequence and tautologies [150]. Historically, this semantic framework is due to Hintikka [82] and is based on Kripke possible worlds semantics [167]. In this framework, an agent is said to know a proposition ϕ if and only if ϕ is true in all possible worlds that are *epistemically accessible* by the agent. The notion of epistemic accessibility is considered primitive and is represented through an accessibility relation on possible worlds, as in Kripke's semantics. From the aforementioned definition of knowledge, it can be easily demonstrated that if an agent knows ϕ then it knows all the logical consequences of ϕ. By classical definition of logical consequence, if ψ is consequence of ϕ, then there are no possible worlds where ϕ is true and ψ false. That is, the set of worlds that satisfy ϕ is a proper subset of the set of worlds that satisfy ψ. Therefore, if ϕ is true in all worlds epistemically accessible to an agent, so is ψ. Similar considerations apply to tautologies: since a tautology is defined as a proposition that is true in all possible worlds, then a tautology is *a-fortiori* true in all the possible worlds that are epistemically accessible to an agent.

Representing knowledge as closed under logical consequence and tautologies is clearly problematic, especially when the focus is on reasoning about real-world agents like humans, computer programs, or AI systems. Consider an agent i that knows Peano's axiom. As long as we represent i's knowledge as closed under logical consequence, we are committed to claim that if i knows Peano's axioms, then i knows all the theorems of arithmetic, which is clearly absurd. Because of logical omniscience, it has been argued that standard epistemic logic[1] has to be regarded not as representing agents' actual knowledge, but rather the information about the world that agents *implicitly* possess given their actual knowledge [109]. This point of view is consistent with several views in early mathematical logic [74, 144, 32, 26], which considered the primary objective of logic not to describe what conclusions real-world agents (humans and machines) are actually able to draw from premises, but to identify which propositions they can ultimately accept as true without becoming incoherent. For the majority of applications of epistemic logic, however, implicit knowledge is almost useless. In fact,

> *"to predict or to explain a real-world agent's actions we need to know what the agent actually knows, and not what it possibly knows"* [54, p. 4]

In other words, what we need is a logic of explicit knowledge. Such a logic should represent agents as neither omniscient nor completely ignorant, i.e., unable to get to know any of the logical consequences of their basic knowledge [54, 136]. What we need is something in between: a logic that represents agents as knowing explicitly all and only

[1]From now on, we will use the term "standard epistemic logic" to refer to all modal systems developed within the model-theoretic tradition and based on Hintikka's modal analysis of knowledge [82]. Notably, these include systems **K**, **T**, **S4**, **S5** and their extensions [167].

the information they can actually infer from their basic knowledge using their limited resources and inferential abilities.

Along the years, several solutions have been proposed for the problem of logical omniscience. Many of them rest within the paradigms of possible worlds semantics [98] and are based on the strategy of weakening epistemic logic so to invalidate axioms and rules responsible for the emergence of logical omniscience [54]. Notable examples include *neighborhood semantics*[2], Levesque's *situations semantics* [109], and various *impossible worlds semantics* [81, 35, 177, 64, 18]. All these solutions are presented more in detail in Section 5.3.1.

As argued in [54, 136], these solutions based on the so-called "strategy of weakening" are mostly unsatisfactory because either do not invalidate all relevant forms of logical omniscience or represent agents as logically ignorant. In both cases, hence, the portrait of the agents they provide remains unrealistic. Indeed, real-world agents are typically neither omniscient nor fully ignorant but *moderately competent*, i.e., able to get to know all and only the information they can actually infer from their actual knowledge using their limited resources and inferential abilities [136].

Other different solutions have been proposed outside the paradigm of possible world semantics. An example is the semantics of *deduction models* proposed by Konolige [95] and inspired by research in logic-oriented AI and expert systems [73]. In this semantics, agents are represented by pairs of a knowledge base KB and a set of rules ρ. The agent knows a certain proposition p if and only if p can be derived from KB through ρ. Depending on the rules that are included in ρ, one can obtain more or less inferentially powerful agents. Logical ignorance corresponds to an empty ρ, while omniscience corresponds to include in ρ all rules of classical propositional logic. Intermediate degrees can be obtained including in ρ only some relevant rules, thus representing agents that are neither omniscient nor ignorant.

More recent examples are represented by dynamic frameworks, e.g., [54, 136]. These extend standard epistemic logic with operators to represent the actions of applying a rule or executing an inference, which are then used to constraint explicit knowledge and invalidate logical omniscience. For example, the $DES4_n$ system proposed in [54] includes an axiom stating that "if an agent (explicitly) knows a proposition ϕ and $\phi \models \psi$, then the agent explicitly know ψ *provided that it performs the right act of thought*". The agent is therefore neither ignorant, as it can get to know some consequences of its actual knowledge, nor omniscient, as its ability to get new knowledge depends on the effective execution of an inference.

Another example is finally provided by depth-bounded logics [59] and their epistemic extensions [103, 29, 104]. These logics constrain the inferential power of an agent by

[2]Alternatively called, minimal-models semantics or Scott-Montague semantics, see [73].

limiting the amount of so-called *virtual information* it can manage, where the latter is represented in terms of the number of arbitrary assumptions an agent can introduce in a logical deduction.

All the mentioned solutions technically resolve logical omniscience insofar they provide a formalism in which both omniscience and ignorance are avoided. However, as we will argue more in details in the following sections, they cannot be regarded as fully satisfactory for a variety of different reasons. One of them is that they do not consider a fundamental ability of rational real-world agents, which is that of *optimizing their resources*. Ideal agents have infinite resources at their disposal, hence they have no necessity of optimising their use. On the contrary, real-world agents have to take care of using their limited resources at best. This is not to say that real-world agents *always* make the best use of their resources, but that they *should* do it if they are rational. As logic is supposed to focus on rational agents, we believe that a good definition of *explicit knowledge* should presuppose that agents are able of making the best use of their limited resources. Following this intuition, we propose a new definition of explicit knowledge according to that:

> *a real-world agent i explicitly knows a proposition ϕ if and only if i is able to get to know ϕ (from its basic knowledge) given its inferential abilities and making the best use of its limited resources.*

Based on this definition, in this chapter we develop a new framework for epistemic logic describing real-world agents' reasoning as a states-space exploration process that can be modelled via Markov decision processes and reinforcement learning. The framework includes: (i) a semantics based on a new class of models called *Markov deduction structures*, (ii) a logic and a dynamic logic of explicit knowledge, and (iii) specific model checking algorithms to verify whether real-world agents' satisfy given desirable epistemic requirements. The chapter is organized as follows. In Section 5.2, we introduce the standard treatment for epistemic logic based on possible world-semantics and discuss its relation with the problem of logical omniscience. In Section 5.3, we discuss some well-known solutions to the problem of logical omniscience proposed in literature, thereby focusing on their strengths and weaknesses. In Section 5.4, we explain how to model real-world agents' reasoning via Markov decision processes whose states are finite sets of formulae and actions correspond to logical rules. Notably, to represent agents with limited resources and abilities, we introduce *Markov deduction structures*, a new class of algebraic structures inspired by Konolige's deduction models, of which the former can be considered a sort of probabilistic generalization. In Section 5.7, we introduce the *Logic of Markov Deduction Structures* (MDSL), a multi-agent logic of explicit knowledge based on the formalism of Markov deduction structures and representing agents as knowing explicitly all and only the information that they can infer from their actual knowledge making the best use of their limited resources and inferential abilities. In Section 5.8, we extend the MDSL to a

dynamic framework to represent exchange of information and resources between agents. In Section 5.9, we introduce specific model checking procedures for both MDSL and its dynamic extension. We conclude with some remarks on future developments, concerning in particular the implementation and the applications of the formalism here presented.

5.2 Epistemic Logic

The standard treatment for the logic of knowledge was originally introduced in [82]. It concerns knowledge as a modal notion, like necessity and possibility, and is largely based on the possible worlds semantics developed by Kripke [98].

Syntax

Let AP be a finite non-empty set of atomic propositions and let $\mathcal{A}$ be a finite non-empty set of agents whose elements are denoted by $i \in \mathcal{A}$. The language of standard epistemic logic is defined as follows:

$$\phi := \quad p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid K^i\phi$$

with $p \in AP$ and $i \in \mathcal{A}$. The language is an extension of standard propositional logic with the addition of a modal operator K^i to represent knowledge of an agent $i \in \mathcal{A}$. In multi-agent contexts, the language is eventually extended with modal operators for *everybody knows* E^Γ, *common knowledge* C^Γ, and *distributed knowledge* D^Γ in a group of agents $\Gamma \subseteq \mathcal{A}$.

Semantics

The standard semantics of epistemic logic is based on Kripke models.

Definition 73 (Kripke model (KM)) *A Kripke model KM is a tuple $\langle \Omega, \{R^i_K\}_{i\in\mathcal{A}}, v\rangle$ where:*

- *Ω is a set of possible worlds;*
- *$\{R^i_K\}_{i\in\mathcal{A}}$ is a family of accessibility relations $R^i_K \subseteq \Omega$ defined on worlds;*
- *$v : \Omega \times AP$ is a Tarskian valuation function on pairs of worlds and atoms such that, for any pair $\langle \omega, p\rangle \in \Omega \times AP$, $V(\omega, p) = 1$ if p is true in ω and $V(\omega, p) = 0$ otherwise.*

Possible worlds are assumed to be coherent and complete with respect to atoms in AP, that is, for all atoms $p \in AP$ and worlds $\omega \in \Omega$, either p is true in ω or it is false. There are neither worlds where p has no definite truth value, nor worlds where p is both true and false.

In the possible-worlds semantics, epistemic logic formulae are typically evaluated with respect to a Kripke model M_K and a world $\omega \in \Omega$, which represents the *actual* world. The satisfiability conditions are defined as follows:

Definition 74 (Formula satisfiability)

$$\begin{array}{lll} M_K, \omega \models p & \text{iff} & v(p, \omega) = 1\,, \\ M_K, \omega \models \neg\phi & \text{iff} & M_K, \omega \not\models \phi\,, \\ M_K, \omega \models \phi_1 \wedge \phi_1 & \text{iff} & M_K, \omega \models \phi_1 \text{ and } M_K, \omega \models \phi_2\,, \\ M_K, \omega \models K^i\phi & \text{iff} & \forall\, \omega' \in \Omega : \omega R^i_K \omega'\ M_K, \omega \models \phi\,. \end{array}$$

Satisfiability conditions for knowledge are analogous to those of necessity and possibility expressed in the possible-world semantics for alethic modal logic. What changes is the significance of the accessibility relation, which is here interpreted as *epistemic accessibility*: we say that a world $\omega' \in \Omega$ is epistemically accessible from ω by an agent i if and only if ω' is coherent (i.e., it does not contradict) with i's actual knowledge at ω.

A Kripke frame (KF) is a class of Kripke models specified by a pair $\langle \Omega, R_K \rangle$ of a set of worlds and an accessibility relation R_K. Different Kripke frames can be obtained by varying the topology of the accessibility relation, e.g. by requiring it to be *reflexive*, *transitive*, or *Euclidean* [167].

Definition 75 (Validity and satisfiability in Kripke frames) *Let $\mathcal{K}$ be a Kripke frame. We say that ϕ is* valid *in $\mathcal{K}$ if and only if ϕ is satisfied by all models in $\mathcal{K}$. We say that ϕ is* satisfiable *in $\mathcal{K}$ if and only if there exists at least one model in $\mathcal{K}$ that satisfies ϕ.*

The largest Kripke frame is usually denoted by $\mathcal{K}$ and defined as the class of all Kripke models that do not impose any restriction on the topology of the accessibility relation. Relevant frames included in $\mathcal{K}$ can be obtained by requiring the accessibility relations to satisfy specific algebraic properties, as follows:

- $\mathcal{T}$ is the class of all Kripke models such that R^i_K for all $i \in \mathcal{A}$ is *reflexive*;
- $\mathcal{S}4$ is the class of all Kripke models such that R^i_K for all $i \in \mathcal{A}$ is *reflexive* and *transitive*;
- $\mathcal{S}5$ is the class of all Kripke models such that R^i_K for all $i \in \mathcal{A}$ is *reflexive*, *transitive*, and *euclidean*;

Notice that, in $\mathcal{S}5$ accessibility relations are in fact *equivalent relations*, hence totally analogous to the relations introduced in the formalism of interpreted systems discussed in the previous chapters. Kripke frames and interpreted systems are intimately related. In particular, an interpreted system can be regarded as an $\mathcal{S}5$-like Kripke frame characterized by a finite set of words Ω, whereas in $\mathcal{S}5$ worlds can be infinite.

5.2.1 Axiomatization

The various Kripke frames introduced above induce a hierarchy of axiomatic systems for epistemic logic. At the basis of the hierarchy we find system K, which is defined by the following axioms and rules:

Definition 76 (System K)

PC	all substitution instances of propositional tautologies ,
K	$\forall\, i \in \mathcal{A}\,,\ K^i(\phi \to \psi) \to (K^i\phi \to K^i\psi)\,,$
MP	from $\phi \to \psi$ and ϕ infer ψ ,
Nec	from the validity of ϕ infer $\forall\, i \in \mathcal{A}\,, K^i\phi$.

The system **K** is complete and coherent with respect to the frame $\mathcal{K}$, i.e., the largest possible class of possible Kripke models. It represents the weakest epistemic logic system insofar all the other axiomatic systems can be obtained as extensions of **K**. Notably, the system **T** is obtained by extending **K** with the *truthfulness axiom*:

$$\mathbf{T} := K^i\phi \to \phi\,,$$

which establishes the truthfulness of knowledge and it is complete and coherent w.r.t. frame $\mathcal{T}$. The system **S4** is obtained by extending **T** with the *positive introspection axiom*:

$$\mathbf{4} := K^i\phi \to K^iK^i\phi\,,$$

and it is complete and coherent w.r.t. the frame $\mathcal{S}4$. Finally, the system **S5** is obtained by extending **S4** with the *negative introspection axiom*:

$$\mathbf{5} := \neg K^i\phi \to K^i\neg K^i\phi\,,$$

and it is complete and coherent w.r.t. frame $\mathcal{S}5$.

The problem of logical omniscience already emerges at the level of system **K** as a consequence of the axiom **K** and the necessiation rule **Nec**, the former inducing closure under logical consequence and the latter inducing closure under tautologies. As all epistemic logic systems include **K**, they are all affected by the problem.

5.3 Solutions to Logical Omniscience

Along the years, several solutions have been proposed for the problem of logical omniscience. These can be divided in three macro-groups: (i) model-theoretic solutions, (ii) syntactic solutions, and (iii) dynamic solutions. In this section, we briefly analyse each of them, identifying their strengths and weaknesses and arguing under which respect they cannot be regarded as satisfactory.

5.3.1 Model-Theoretic Solutions

These solutions are based on weakening standard epistemic logic, invalidating the axioms and rules responsible for the emergence of logical omniscience [54]. Arguably the most representative kind of model-theoretic solutions are based on *neighborhood semantics*[3], Levesque's *situations semantics* [109], or *impossible worlds semantics* [18].

Neighborhood Semantics

Given a set of possible worlds Ω, a *neighborhood function* N is a mapping $N : \Omega \to 2^{2^{\Omega}}$ that assigns to each $\omega \in \Omega$ a set of sets of possible worlds. A neighborhood model M_N is a tuple $\langle \Omega, N, v \rangle$ of a set of possible worlds Ω, a neighborhood function N, and a valuation function v. A semantics for epistemic logic can be formulated in terms of neighborhood models by including a neighborhood function N^i for each agent $i \in \mathcal{A}$ and then establishing the following condition:

$$M_N, \omega \models K^i\phi \text{ iff } Sat(\phi) \in N^i(\omega)\,,$$

where $Sat(\phi)$ is the set of all worlds that satisfy ϕ. That is, an agent knows a given proposition ϕ if and only if the set of worlds that satisfy ϕ are among those associated to ω by the neighborhood function N^i. As there are no constrains on which sets of worlds N^i assigns to a given world ω, closure under logical consequence is no longer valid. For example, let assume that $\Omega := \{1, 2, 3\}$ and consider two propositions ϕ and ψ such that $Sat(\phi) := \{2\}$ and $Sat(\psi) := \{2, 3\}$. In this frame, it holds that $\phi \models \psi$ as there are no worlds ω where ϕ is true and ψ is not. Let assume that 1 is the actual world and that $K^i\phi$. In terms of neighborhood semantics, this is equivalent to say that $\{2\} \in N^i(\omega)$. Since N^i is completely arbitrary, from $\{2\} \in N^i(\omega)$ does not follow that also $\{2, 3\} \in N^i(\omega)$. Therefore, we can have models in which $K^i\phi$ holds and $K^i\psi$ does not hold even if $\phi \models \psi$. Closure under logical consequence is thus invalidated. A similar reasoning holds for closure

[3] Alternatively called, minimal-models semantics or Scott-Montague semantics, see [73].

under tautologies. To prevent an agent i from knowing all tautologies in a given world ω, it is in fact sufficient to not include Ω in $N^i(\omega)$.

The complete and coherent epistemic axiomatization of neighborhood semantics corresponds to the minimal system with no axioms in addition to propositional tautologies and two rules: *modus ponens* and the following *equivalence rule*:

$$\text{from } \phi \leftrightarrow \psi \text{ and } K^i\phi \text{ infer } K^i\psi\,.$$

Neither the axiom **K** nor the rule **Nec**, responsible for the emergence of logical omniscience, are present.

However, Neighborhood semantics and its axiomatization rise two fundamental issues making them unsatisfactory solutions. The first one is that agents may potentially not know *any* logical consequence of their actual knowledge, i.e., they could be completely ignorant. This is a direct consequence of the fact that N^i is completely arbitrary and does not induce any constraint (in principle) on agents' knowledge. Of course, one could act on the specification of N^i in order to include some relevant constraints. For example, one could specify N^i in such a way that if $Sat(\phi) \in N^i(\omega)$ for a given ω, then, for all propositions ψ that can be derived from ϕ, say, within seven applications of modus ponens, it holds that $Sat(\psi) \in N^i(\omega)$. However, this solution looks *ad-hoc* and quite cumbersome to implement in practice.

Another issue rises from the fact that neighborhood semantics does not invalidate all relevant forms of logical omniscience. Indeed, agents are omniscient with respect to a special form of logical consequence, that is, *logical equivalence*. Consider two propositions ϕ and ψ. Let assume that $Sat(\phi) \in N^i(\omega)$. If ϕ and ψ are logically equivalent (i.e., $\phi \leftrightarrow \psi$), then it holds that $Sat(\phi) = Sat(\psi)$. Therefore, if $Sat(\phi) \in N^i(\omega)$, then also $Sat(\psi) \in N^i(\omega)$ and, consequently, by the semantics of the knowledge operator, if $K^i\phi$ and $\phi \iff \psi$, then $K^i(\psi)$. Notice that, in the axiomatization of neighborhood semantics, this is precisely what is established by the equivalence rule. In conclusion, neighborhood semantics fails both to avoid logical ignorance and to invalidate all relevant forms of logical omniscience. Indeed, although weaker than closure under logical consequence and tautologies, closure under logical equivalence is still problematic for applications and leads to several counter-intuitive conclusions. For example, suppose that an agent i knows that $2 + 2 = 4$, since $2 + 2 = 4$ is logically equivalent to all theorems of arithmetic, if we represent i's knowledge in terms of neighborhood semantics, we are forced to say that i knows all theorems of arithmetic, which, again, is absurd.

Notice that closure under logical equivalence represents an issue not only for neighborhood semantics but for all semantic approaches based on the paradigm of classical possible worlds insofar they assume each *proposition* to be equivalently representable via its *truth-conditions* and, thus, via the *sets of possible worlds satisfying it*. For all the semantic approaches assuming this equivalence (i.e., all the approaches based on the paradigm of

classical possible worlds), there is no way to distinguish between propositions being true in the same class of possible worlds: if two propositions have the same truth-conditions they in fact count as *the same proposition*, hence the closure under logical equivalence.

In most contexts, included alethic modal contexts, this form of closure is not problematic as the *salva-veritate* substitutability holds: two components of a statement having the same truth conditions can be replaced with one another without altering the statement's truth-value. For example, in the sentence "necessarily $2 + 2 = 4$ holds", one can replace the proposition "$2+2 = 4$" with another having the same truth-condition (e.g., "Fermat's Theorem") and the truth-value of the whole sentence remains the same. This is not the case of epistemic contexts, where replacing a proposition with another having the same truth conditions can alter the overall truth value of the epistemic statement. By way of exmaple, consider again the two propositions "$2+2 = 4$" and "Fermat's theorem". Even if they have the same truth-conditions, we cannot replace them with one another in an epistemic statement as we do in other modal contexts. The statements "Alberto knows that $2 + 2 = 4$" and "Alberto knows Fermat's theorem", indeed, *are not logically equivalent*: there may be cases in which one is true and the other false.

From what said, it follows that closure under logical equivalence emerges whenever one adopts a semantic approach based on classical possible worlds, hence making it impossible to elaborate a satisfactory solution to the problem of logical omniscience while remaining within this paradigm. Various solutions have therefore been proposed outside the paradigm of classical possible worlds, but always remaining within the semantic model-theoretical tradition. The latter are generally based on the assumption of the existence of *non-classical possible worlds*, such as Levesque's *situations*, which will be discussed in the following paragraph, and *impossible worlds*, which we will examine in the next one.

Situations Semantics

A different model-theoretic solution that goes beyond the paradigm of classical possible worlds is the *situations semantics* proposed by Levesque [109]. Situations are *"generalizations of possible worlds where not every sentence in a language is required to have a truth value"* [109, p.199]. Models M in situations semantics are defined as tuples $\langle S, B, \mathbf{T}, \mathcal{F}\rangle$ where S is the set of all situations, B is the set of situations that could be the actual one according to what is believed, $\mathcal{T} : AP \to 2^S$ is a function that assigns to each atom p of the language the set of situations $\mathcal{T}(p)$ that supports the truth of p, and $\mathcal{F} : AP \to 2^S$ is a function that assigns to each $p \in Ap$ the set of situations $\mathcal{F}(p)$ that supports the falsity of p. The set of possible worlds Ω is then defined as the set of all situations $s \in S$ such that, for all $p \in AP$, either $s \in \mathcal{T}(p)$ or $s \in \mathcal{F}(p)$, i.e., a possible world is a situation that assigns a truth-value to each atoms in the language. The set of possible worlds compatible with a given situation $s \in S$, denoted $\Omega(s)$, is defined as the set of all worlds $\omega \in \Omega$ such

that: (i) if $s \in \mathcal{T}(p)$ then so is ω, and (ii) if $s \in \mathcal{F}(p)$ then so is ω. That is, a world ω is compatible with a given situation s if and only if ω completes s, in the sense that it assigns a definite truth-value to all atoms that have no definite truth-value in s.

Levesque uses situations semantics to distinguish between *implicit* and *explicit* knowledge. The former is modelled as in standard possible-world semantics and is closed under logical consequence and tautologies. The latter represents what a real-world agent *actually* knows. A specific operator E is introduced in the language to model explicit knowledge whose semantics is as follows:

Definition 77 (Explicit knowledge (Levesque)) *Given a model M and a situation $s \in S$, the following condition holds:*

$$M, s \models E\phi \text{ iff } \forall s' \in B : s' \in \mathcal{T}(p)$$

The incompleteness of situations prevents explicit knowledge to be closed under logical consequence and tautologies. For example, consider two atoms p and q and suppose that $p \models q$ and $E(p)$. In situations semantics, this corresponds to say that $\{\omega : \omega \in \mathcal{T}(p)\} \subseteq \{\omega : \omega \in \mathcal{T}(q)\}$ and that $\forall s' \in B : s' \in \mathcal{T}(p)$. In this scenario, there may be situations in B that assign value true to p (i.e., they are in $\mathcal{T}(p)$) but do not assign any truth-value to q (i.e., they are neither in $\mathcal{T}(p)$ nor in $\mathcal{F}(p)$). Notice that these situations are perfectly compatible with the condition that $\{\omega : \omega \in \mathcal{T}(p)\} \subseteq \{\omega : \omega \in \mathcal{T}(q)\}$, i.e., with the requirement that if a situation s is in $\mathcal{T}(p)$, then all worlds that complete s assign the truth-value "true" to q. However, if some s exists that assigns truth-value "true" to p but does not assign any truth-value to q, then $M, s \not\models E(q)$. Hence, from $p \models q$ and $E(p)$ does not follow $E(q)$. Closure under logical consequence is therefore violated, as desired. A similar reasoning holds for closure under tautologies. From the fact that a proposition p is true in all possible worlds (i.e., $\forall \omega : \omega \in \mathcal{T}(p)$) it does not follow that for all situations p is also true, simply because there may be situations in which p is neither true nor false.

Situations semantics invalidate all relevant forms of logical omniscience, including closure under logical equivalence that remains instead valid neighborhood semantics. For the latter, simply observe that if two propositions p and q are true in exactly the same class of possible worlds (i.e., they are logically equivalent), there may nevertheless be situations in which p is true and q has no defined truth-value, and vice versa. Unfortunately, on the other hand, situation semantics suffer from the same problem that makes neighborhood semantics unsatisfactory, i.e., it represents agents as *logically ignorant*. As neighborhood models, indeed, situations do not impose any constraint on the agents' knowledge, i.e., there is nothing in the semantics granting that certain logically consequences can be known if the agent has enough competences and resources, unless to introduce complicate *ad-hoc* restrictions (e.g., by imposing that for all propositions ϕ that can be derived by a set of

premises Γ via at most four iterated applications of *modus ponens*: if all situations in B assign value "true" to all propositions in Γ, then they also assign value "true" to ϕ). In our research of a threshold between omniscience and ignorance, therefore, situations semantics does not qualify as a satisfactory solution.

Impossible Worlds

The last model-theoretic approach we examine is that based on *impossible worlds semantics*. This approach has been explored by a variety of authors in recent years, including Hintikka [81], Cresswell [35], Wansing [177], Fagin [64], and Berto et Jago [18]. In a nutshell, it consists of extending Ω with a set of impossible worlds $\Omega^{\perp}$ where rules of classical logic do not hold. Truth in impossible worlds is modelled through a valuation function $\delta : \mathcal{L}_{AP} \rightarrow [0, 1]$ that assigns to each formula of the language $\mathcal{L}_{AP}$ and impossible world $\omega^{\perp} \in \Omega^{\perp}$ an arbitrary truth-value. Impossible-worlds do not behave classically. For example, we can have impossible worlds that satisfy both ϕ and ψ but do not satisfy $\phi \wedge \psi$.

In impossible worlds semantics, the knowledge operator is evaluated with respect to both possible and impossible worlds. Let $\Omega^{+} := \Omega \cup \Omega^{\perp}$ and let $\mathcal{A}$ be a finite non-empty set of agents, a model M in impossible worlds semantics is defined as a tuple $\langle \Omega^{+}, \mathcal{A}, \{R^i\}_{i \in \mathcal{A}}, v, \delta \rangle$, with v being the usual valuation function, δ being the valuation function for impossible worlds and $\{R^i\}_{i \in \mathcal{A}}$ being a family of accessibility relations defined on both possible and impossible worlds. The semantics of the knolewdge operator K^i is then defined as follows:

Definition 78 *Given a model M and a world $\omega^{+} \in \Omega^{+}$, the following condition holds:*

$$M, \omega^{+} \models K^i \phi \;\text{ iff }\; \forall\, \omega' \in \Omega^{+} \,\text{s.t.}\, \omega R^i \omega' :\; M, \omega' \models \phi$$

Since knowledge is evaluated considering also impossible worlds, which are completely ill-behaved, logical omniscience does not hold. To invalidate closure under logical consequence, it is sufficient to have an impossible world among those accessible in which ϕ and $\phi \rightarrow \psi$ hold but ψ does not hold. Similarly, closure under a given tautology can be easily invalidated by including an impossible world in which that tautology does not hold.[4]

5.3.2 Syntactic Approaches

Outside the model-theoretical framework, a completely innovative approach to logical omniscience is the "syntactic" approach proposed by Konolige in [95] and based on *deduction*

[4]Notice that the definition of tautology is the same as in classical logic, a proposition ϕ is a tautology if and only if it is true in all *possible* worlds, hence excluding the impossible ones.

models, formal representations of agents inspired by logic-based experts systems. A deduction model d^i for a given agent $i \in \mathcal{A}$ is a pair $\langle KB, \rho \rangle$ where KB is a finite set of formulae in propositional language and ρ is a set of logical rules that apply to formulae in KB. ρ is not required to include all rules of propositional logic and may be incomplete. We write $\vdash_i \phi$ to denote that formula ϕ is *derivable* in the deduction model of i, i.e., it can be inferred from KB by iterative applications of rules ρ.

In Konolige's framework, a model M is defined as a tuple $\langle \Omega, \{d^i\}_{i \in \mathcal{A}}, v \rangle$ with Ω a set of possible worlds, $\{d^i\}_{i \in \mathcal{A}}$ a family of deduction models, one for each agent $i \in \mathcal{A}$, and v the valuation function. The semantics of the propositional fragment is formulated with respect to possible worlds, as in standard epistemic logic. The knowledge operator K^i, instead, is interpreted on deduction models as follows:

Definition 79 *Given a model M, the following definition holds:*

$$M \models K^i\phi \text{ iff } \vdash_i \phi\,.$$

That is, an agent knows a given proposition ϕ if and only if ϕ is derivable in the deduction model associated to i.

Concerning the proof theory, the system BK provides a complete and coherent axiomatization of the semantics of deduction models. In addition to all axioms and rules of propositional logic, it includes: (i) a set ρ of rules for each agent $i \in \mathcal{A}$, and (ii) the *procedural link* rule stating that:

- from $K^i\phi_1 \ldots K^i\phi_n$ and $\phi_1, \ldots, \phi_n \vdash_i \psi$, it follows that $K^i\psi$.

In practice, each agent $i \in \mathcal{A}$ is represented as knowing all and only the propositions that it can derive from KB through rules ρ. Agents in the logic of deduction models are thus represented as neither omniscient nor ignorant. In this regard, Konolige's framework provides an effective solution to the problem of logical omniscience. In our opinion, however, this solution is not really satisfactory for a variety of reasons. First of all, Konolige's framework assumes the lack of logical omniscience to depend only on agents' incomplete knowledge of classical logic rules, while it depends also (and even mostly) on their limited computational resources. Second, although knowledge is not closed under logical consequence, it is still deductively closed with respect to rules in ρ. In some cases, this form of closure may be unrealistic. Consider for example a proposition ϕ that the agent i can infer from KB only through an incredibly high number of iterated applications of rules ρ: would we still be prepared to say that i knows ϕ? if i has no sufficient time or resources, it would never be able to know ϕ, even if the latter is derivable in its deduction model. Third, the fact that an agent knows all classical logic rules clearly does not imply that it is logically omniscient: there are plenty of real-world agents that know classical logic rules

and Peano's axioms but none of them knows all theorems of arithmetic, simply because they have not enough time and resources to derive them.

In conclusion, the logic proposed by Konolige solves logical omniscience but adopting a wrong strategy. In particular, its main failure relies on not taking explicitly into account agents' resources and how these are used. Nevertheless, this logic makes a fundamental progress with respect to model-theoretic solutions: relating knowledge explicitly to the inferential procedures occurring in agents' memory is very valuable and represents the starting point of our framework, which is largely inspired by Konolige's proposal.

5.3.3 Dynamic Approaches

We conclude by analysing solutions based on dynamic epistemic logic [168]. As noticed in [136, p.9], it is *"an agent's ability to perform bounded, but non-trivial, logical reasoning that makes agents logically non-omniscient and logically non-ignorant at the same time"*. Focusing on the relationship between knowledge and reasoning is therefore fundamental to reach a good solution to the problem of logical omniscience. A way to model this relationship is via Konolige's deduction models presented in the previous section, another is by introducing specific dynamic operators to represent agents' reasoning processes. This is the strategy explored by a number of dynamic theories of knowledge proposed in literature, such as the *Active Logics* [60], the *Logic of Finite Syntactic Epistemic States* proposed in [2] and Duc's dynamic epistemic logic proposed in [53] and further developed in [54, 55]. By way of example, we discuss here the dynamic epistemic logic $\mathbf{L}_D$ proposed by Rasmussen in [136], which can be considered itself and advancement over previous proposals.

Let $\mathcal{R}$ be a finite, possibly empty set of inference rules that an agent knows and can apply. Let $R_i \in \mathcal{R}$ denote a generic element of this set. Let $\lambda_i \in \mathbf{R}$ denote the cognitive cost of applying the rule R_i. The language of $\mathbf{L}_D$ is defined as follows:

$$\phi := p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid K \mid \langle R_i \rangle^{\lambda_i}\phi \mid [R_i]^{\lambda_i}\phi$$

It includes atoms p, standard Boolean connectives for negation and conjunction, the standard knowledge operator K and two dynamic operators with the following reading:

- $\langle R_i \rangle^{\lambda_i}\phi$ means: "after some application of the inference rule R_i at cognitive cost λ_i, then ϕ is the case";
- $[R_i]^{\lambda_i}\phi$ means: "after any application of the inference rule R_i at cognitive cost λ_i, then ϕ is the case".

Arbitrary sequences of dynamic operators can be represented through the following notation:

$$\begin{aligned} \langle \Re \rangle^{\lambda} &:= \quad \langle R_i \rangle^{\lambda_i}, \ldots, \langle R_i \rangle^{\lambda_j} \,, \\ |\Re|^{\lambda} &:= \quad [R_i]^{\lambda_i}, \ldots, [R_j]^{\lambda_j} \,, \end{aligned}$$

where $R_i, \ldots, R_j$ are arbitrary inference rules and $\lambda = \lambda_i + \cdots + \lambda_j$.

Contrary to most of the solutions presented so far, the solution outlined in [136] is presented in proof-theoretic rather then in semantic terms [5]. The author provides an Hilbert-style axiomatic system including the rule *modus ponens* and the following schema of axioms:

Definition 80 (Axioms of $\mathbf{L}_D$)

PC	all substitution instances of propositional tautologies ,
A1	$\langle \Re \rangle^{\lambda} K^i \phi \rightarrow \phi \,,$
A2	$\langle \Re \rangle^{\lambda} K^i \phi \rightarrow \langle \Re \rangle^{\lambda} [\Re]^{\lambda} K^i \phi \,,$
A3	$\langle \Re \rangle^{\lambda} \phi \wedge \langle \Re' \rangle^{\lambda'} \psi \rightarrow \langle \Re \rangle^{\lambda} \langle \Re' \rangle^{\lambda'} (\phi \wedge \psi) \,,$
A4	$\langle \Re \rangle^{\lambda} (\phi \wedge \psi) \rightarrow \langle \Re \rangle^{\lambda} \phi \,,$
$\mathbf{R_D}$	$\langle \Re \rangle^{\lambda} (K^i \Gamma) \wedge (\Gamma \rightarrow_{R_i} \phi) \rightarrow \langle \Re \rangle^{\lambda} \langle R^i \rangle^{\lambda_i} K^i \phi$

$A1$ is the analogous of the factivity of knowledge in standard epistemic logic. It establishes that only true propositions can be inferred through rules. $A2$ establishes that known propositions remain known through reasoning. $A3$ establishes that if two propositions ϕ and ψ can be derived through two independent processes of reasoning $\langle \Re \rangle^{\lambda}$ and $\langle \Re' \rangle^{\lambda'}$, their conjunction $\phi \wedge \psi$ can be derived through the subsequent execution of $\langle \Re \rangle^{\lambda}$ and $\langle \Re' \rangle^{\lambda'}$. $A4$ establishes that if a conjunction $\phi \wedge \psi$ can be derived through rules $\langle \Re \rangle^{\lambda}$, then it so one of its conjunct. Finally, R_D is a schema including a specific axiom for each rule R_i known by an agent. Each one of these axioms establishes that, given a rule R_i, if i knows R_i and enough resources to apply it, then i knows all the propositions that can be derived from its basic knowledge via one-step application of R_i.

By means of $\mathbf{L}_D$, we can represent agents that are neither completely ignorant nor omniscient. In this regard, $\mathbf{L}_D$ provides an effective solution to the problem of logical omniscience. To prove this result, we can proceed as in [136, sec. 3.3]. For what concerns ignorance, we have two cases. When $\mathcal{R} \neq \emptyset$, agents can get to know new propositions via rules application and thus are clearly not ignorant. When $\mathcal{R} = \emptyset$, agents are in fact ignorant but, since we are explicitly assuming that they know no rules of inference (i.e., they have no inferential abilities at all), this is not paradoxical. For what concerns omniscience, it can be proved showing that neither closure under logical consequence nor closure under tautologies hold in the axiomatization of $\mathbf{L}_D$. For the former, it is sufficient to notice that $\mathbf{L}_D$ does not prove the following sequence:

[5] Even if the author promises to develop a model-theoretic semantics for the proposed logic in the future, see [136, p. 2].

$$CL := \; K\Gamma \wedge (\Gamma \rightarrow_R \phi) \rightarrow K\phi \,,$$

where R is any relevant law of logic.

To prove that CL does not follow from the axioms of $\mathbf{L}_D$, it is sufficient to show that it explicitly violates the schema $\mathbf{R}_D$. For CL, if an agent knows Γ and $\Gamma \rightarrow_R \phi$, then the agent automatically knows ϕ. By contrast, for $\mathbf{R}_D$ the fact that the agent knows Γ and $\Gamma \rightarrow_R \phi$ are not sufficient to state that the agent knows ϕ. It is also fundamental that the agent knows the relevant rule R and is able to apply it given its cognitive cost. For this to happen, two conditions are necessary: (i) $R \in \mathcal{R}$, and (ii) the agent's resources have to be sufficient to support the cognitive cost of R.

Compared to Konolige's syntactic approach, which allows to derive the lack of logical omniscience of real-world agents from the incompleteness of their deductive apparatus, the dynamic solution presented in [136] takes in consideration also another fundamental reason why real-world agents are not omniscient: their limited resources. In this regard, the logic $\mathbf{L}_D$ satisfies the desired requirement of representing agents as knowing all and only the propositions that can be derived from their basic knowledge given their limited resources and inferential abilities. However, it still has some limitations, for what concerns our purposes. First of all, it forces the representation of agents' reasoning processes into syntax, where these should be more properly represented in the semantics. Statements of epistemic logic have the task of describing agents' epistemic states and their logical relationships, not their conditions of possibility. The latter, including the reasoning processes that support the possibility of the occurrence of given epistemic states, should be more properly represented in the semantics. Second, this type of solution is not very suitable for model checking or formal verification tasks. The absence of a proper semantic framework to model agents and their reasoning behaviour makes it difficult to apply model-checking techniques, while the language of $\mathbf{L}_D$ is too complex to be suitable for property specifications. Third and most important, $\mathbf{L}_D$ does not consider a fundamental aspect of reasoning with limited resources, that is *optimization learning*. Real-world agents do not apply rules randomly but try to find the pathway that allow them to minimize the use of resources. This fundamental aspect, which is not considered in none of the solutions presented so-far, will instead represent the starting point of our framework based on Markov decision processes and reinforcement learning.

5.3.4 Depth-Bounded Logics

The final approach to logical omniscience we analyse before to introduce our framework is that of *Depth-Bounded Logics*, whose more mature formulation can be found in [59]. Differently from the others here presented, this approach is not tailored specifically on epistemic logic but focuses directly on the notion of *logical consequence* beyond all classical

logic formalisms, from propositional logic to its various extensions. Proponents of depth-bounded logics, indeed, bring the origin of the problem of logical omniscience back to the *received view* of logical empiricism according to which logical statements are "analytical truths", thus trivial and uninformative by their very nature. Logical reasoning, empiricists claim, is not amplifying insofar all the information that we find in the conclusions of a logical deduction is already contained in its premises; the logician's work only consists of making them explicit. This idea is sharply in contrast with computational complexity results proving logical-deductive reasoning to be computationally hard. First-order logic, just to given an example, is notoriously *undecidable*, which means that we have no finite computational procedures to decide whether a statement expressed in this logic is valid or not (i.e., whether a certain consequence follows from a given set of premises). For what concerns propositional logic and its extensions, the decision problem can be solved but its complexity is co-NP complete [31], meaning that it is among the most difficult problems that can be solved via a finite computational procedure. Things do not go better for many sub-classical logics: intuitionistic logic, for example, is P-SPACE complete, meaning that its complexity is likely harder then that of classical propositional logic [151], while the three most famous systems of relevance logics (i.e., *Entailment*, *Relevance Logic*, and *Ticket Entailment*) are all proved to be undecidable [166]. Therefore, how is it possible that logical reasoning is, at the same time, trivial and computationally hard? According to proponents of Depth-bounded logic [58], the answer is that, contra the received-view, logical statements are not (always) analytic and thus logical reasoning is not (always) trivial. The idea of the analyticity of logic, they argue, was introduced in the received view by logical empiricists motivated by their rejection of the Kantian notion of *synthetic a-priori*, which they considered incompatible with their empiricist epistemology and the scientific view of the world. In the depth-bounded logics paradigm, the Kantian idea of *synthetic a-priori* is rehabilitated through the development of an informational semantics and the introduction of the related concept of *virtual information*. The latter is information (additional to that included in the premises Γ) that an agent must assume in order to derive a certain logical consequence ϕ of Γ, and eventually discharges only after proving that $\Gamma \vdash \phi$. Virtual information therefore plays a crucial role in logical deduction. Consider for example the *disjunction elimination* rule in a natural deduction system for propositional logic: the latter asserts that given a disjunction $\phi_1 \vee \phi_2$, if one is able to prove that ψ follows independently from both ϕ_1 and ϕ_2, then one can conclude that ψ follows from $\phi_1 \vee \phi_2$ regardless of which among ϕ_1 and ϕ_2 is true. The application of this rule requires both the disjoints ϕ_1 and ϕ_2 to be assumed at different steps of the derivation, and eventually discharged only after proving that ψ follows independently from both of them. Notice that the steps of assuming the truth of ϕ_1 and ϕ_2 are "non-analytic" as they require one to go beyond the information it actually possesses and simulate scenarios in which it holds information that it actually does not hold [58, pag. 23]. The necessity of managing virtual

information is what makes logical reasoning computationally hard and so it explains why real-world agents are not logically omniscient. Stated otherwise, each further introduction of virtual information in the course of reasoning involves a computational cost for the agent that erodes its resources, hence limiting the number of conclusions it is able to derive from a given set of premises.

To obtain a logic closer to real-world agents' reasoning abilities, supporters of the depth-bounded approach thus propose to limit the amount of virtual information an agent can introduce in the course of a derivation. In technical terms, this result is obtained by formulating a proof-theory consisting of classical *intelim* rules for connectives and a *cut* rule BP (for *Bivalence Principle*), which corresponds to branching the deduction into two paths, one following from assuming the virtual piece of information ϕ, and the other following from assuming $\neg\phi$ (see, Figure 5.1). A *depth-bounded* derivability relation $\vdash_k$

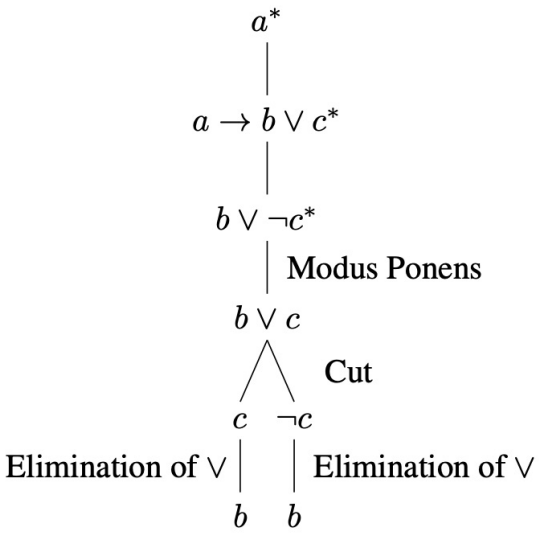


Figure 5.1: Example of a derivation involving the cut rule. Image borrowed from [38].

(with $k \in \mathbb{N}$) is then introduced, where $\Gamma \vdash_k \phi$ can be read as "ϕ is derivable from Γ by applying at most k instances of BP (i.e., by introducing virtual information at most k times). For $k \geq 0$, we obtain a hierarchy of depth-bounded tractable approximations of propositional logic. The basis of this hierarchy is given by setting $k = 0$. The latter corresponds to a proof-system including only the intelim rules and not BP. This 0-depth system does not require to introduce any amount of virtual information and is thus purely analytical and computationally trivial. On the other hand, classical propositional logic corresponds to $\lim_{k\to\infty}$, while more or less powerful system can be obtained by fixing the depth k, that is, by fixing the maximal amount of virtual information an agent can manage.

The interesting result is that all the depth-bounded approximations of the hierarchy are computationally *tractable*, i.e., for each $k \geq 0$, the complexity of deciding whether or not a formula is derivable at depth k is polynomial to the lenght of the formula[6] [59]. This result provides a further reason in support of the idea that the correct strategy to obtain a logic for real-world agents consists in limiting the amount of virtual information that the agent can handle.

Regarding the semantics, for the 0-depth system this is given in terms of the three-valued *non-deterministic* truth-tables reported in Figure 5.2. The latter provides an *in-*

θ	ϕ	$\theta \wedge \phi$	$\theta \vee \phi$	$\theta \rightarrow \phi$	$\neg\theta$
0	0	0	0	1	1
0	1	0	1	1	1
0	$\bot$	0	$\bot$	1	1
1	0	0	1	0	0
1	1	1	1	1	0
1	$\bot$	$\bot$	1	$\bot$	0
$\bot$	0	0	$\bot$	$\bot$	$\bot$
$\bot$	1	$\bot$	1	1	$\bot$
$\bot$	$\bot$	$\bot, 0$	$\bot, 1$	$\bot, 1$	$\bot$

Figure 5.2: Depth-bounded Logics Truth-tables. Image borrowed from [38].

formational reading of truth-values such that:

- $v(\phi) = 1$ means "the agent holds the information that ϕ is true";
- $v(\phi) = 0$ means "the agent hold the information that ϕ is false";
- $v(\phi) = \bot$ means "the agent does not hold any information about the truth-value of ϕ" (here the notation $\bot$ must not be confused with the usual logical constant for "falsum").

Non-determinism arises when some of the two arguments of a binary connective (respectively $\wedge$, $\vee$, and $\rightarrow$) takes value $\bot$. In this case, the truth-value of the whole formula depends on additional information that the agent may or may not hold. Consider for example the disjunction $\phi_1 \vee \phi_2$ and suppose that $v(\phi_1) = v(\phi_2) = \bot$. In this case, the truth-value of $\phi_1 \vee \phi_2$ will depend on whether or not the agent holds the information that ϕ_1 and ϕ_2 may not be simultaneously false. If the agent holds such information, then

[6] measured in terms of the number of literals it includes.

$v(\phi_1 \vee \phi_2) = 1$ (even if the agent does not know which of the two disjoints is true), otherwise $v(\phi_1 \vee \phi_2) = \bot$. Borrowing the famous example by Quine [133], we may surely deny the conjunction "animal X is a mouse *and* animal X is a chipmunk" even if we neither hold the information that "animal X is a mouse" nor we hold the information that "animal X is a chipmunk". The corresponding 0-depth satisfiability relation $\models_0$ is defined as usual by requiring that all the valuations satisfying the premises also satisfy the conclusion.

So far the majority of works on depth-bounded logics available in literature focuses on propositional and first order logic. As such, they cannot strictly be proposed as solutions to logical omniscience as they lack of both a notion of *agent's knowledge* and an agent-relative ("subjective") interpretation of the depth-parameter k. Actual formulations of depth-bounded logics (see, [59]) present the depth k as a measure of the *objective difficulty* of an inference, whereas in epistemic contexts we would k to represent a measure of the subjective *computational/cognitive power* of an agent. In this regard, a preliminary extension of the depth-bounded logic framework to epistemic contexts has been recently presented by Larese in her PhD thesis [103]. The author focuses on multi-agent settings and considers three infinite hierarchies of systems, respectively **DBELu**, **DBELe**, and **DBELc**. Each of these hierarchy includes infinite approximations $\mathbf{DBELx}_k$ (with $x = u, e, c$) for increasing values of k. The difference between the hierarchies depend on background assumptions about what agents know about the agent's depth k of others agents [103, p. 235]:

- In **DBELu**, none of the individuals knows that every individual's depth is at least k, for some fixed $k \in \mathbb{N}$;
- In **DBELe**, every individual knows that every individual's depth is at least k, for some fixed $k \in \mathbb{N}$;
- In **DBELc**, it's common knowledge that every individual's depth is at least k, for some fixed $k \in \mathbb{N}$.

The language is common in all these three hierarchies and is the same of standard epistemic logic, a part for the fact that the knowledge operator K^i_k, which is labelled by an index k denoting agent i's depth. Semantically, each agent $i \in \mathcal{A}$ in a model M is paired with a valuation function δ^M_i that models the information actually available to the agent. That is, if i actually holds the information that ϕ is true (respectively, false), then $\delta^M_i = 1$ (respectively, $\delta^M_i = 0$). Otherwise, $\delta^M_i = \bot$. The set of information actually available to the agent i in the model M can be equivalently denoted by δ^M_i. Hence, we have that $M \models K^i_k\phi$ holds if and only if $\delta^M_i \vdash_k \phi$, i.e., the agent i knows ϕ at a fixed depth k if and only if i can derive ϕ from its actual information by introducing at most k instances of virtual information. Similar semantics are introduced for the operators

of everybody knows, common knowledge, and distributed knowledge, where the specific satisfiability conditions vary depending on the hierarchy considered. For more details on this topic, we refer to [103, Sec. 6.2].

The framework proposes by Larese and based on depth-bounded logics provides an effective solution to the problem of logical omniscience, notably insofar it is able to represent agents as knowing all and only the propositions that they can infer given their limited resources and inferential abilities. For this claim, it is sufficient to note that the agent's depth k, which constraints agent's knowledge, is in fact a measure of the latter's resources and abilities. However, we identify two major limitations in the formalism proposed by Larese and, more in general, in the approaches based on depth-bounded logics. The first limitation is given by the assumption that limitations in agents' inferential power only depends on the amount of virtual information they are able to manage. This assumption implies that two inferences requiring the same amount of virtual information comport the same cost for the agent. Furthermore, it also implies that PB is the only rule that has an effective cost for the agent, while all the other intelim rules have cost zero. This sounds quite as an idealization, which is also in contrast with empirical evidence showing not only that even the most simple rules (like *modus ponens*) may have a cost for real-world agents, but also that different agents potentially incur in different costs for the same rules. This limitation, we believe, is due to the shift that Larese does from the *objective* notion of depth to the *subjective* notion of agent's depth. This apparently trivial conceptual shifts actually implies to assume that an objective measure, introduced to account for the computational properties of deductive inferences, can be used to characterize a subjective (and only empirically measurable) property like the inferential power of agents. Such an assumption is arguably wrong and leads to the aforementioned issue. In addition to this limitation, Larese's framework, like all those analyzed so far, does not take into account the ability of agents to learn and optimize their resources, which we consider fundamental to properly account for real-world agents abilities. Despite these limitations, Larese's framework captures some fundamental intuitions that we will use later in the development of our proposal. For this reason, together with Konolige's deduction models, it represents one of two starting points of our work.

5.4 A New Framework

In this section, we present our proposal. Before presenting the details, we need to briefly recap the fundamentals of Markov decision processes (see, [10]) and reinforcement learning (see, [155]). For the former, we refer to Chapter 1, Section 1.5.4. An introduction to so-called *model-free* reinforcement learning, on the other hand, is outlined in the following section.

5.4.1 Reinforcement Learning

Reinforcement Learning (RL) is a Machine Learning technique that aims to learn from interaction of the agent with the external environment, and is inspired from behaviorism in that it maximizes the *reward* obtained by the agent throughout such an interaction [155]. In a nutshell, Reinforcement Learning usually builds upon the idea of modeling the agent and the environment as a Markov Decision Process which describe the control problems we are interested in, and it aims at finding the best policy, *even when the MDP is not fully known or specified*. One way of ranking policies according to the expected reward they yield is described in the previous section.

In this section we focus on *model-free* RL, which works under the aforementioned condition in which the MDP cannot be fully specified. This means the agent does not have any way of predicting the result of its actions *a priori*, and can only proceed by trial and error. In absence of a model, we can let the agent explore and evaluate the space of policies by trial and error. For instance, a strategy may be that of starting with some random policy and follow it until the end of the *episode* (i.e., an absorbing state is reached). More precisely, we can let the agent start from the initial state s_0, perform an action according to the policy, and get a reward and the new state from the environment. At this point, the agent picks another action from the policy, and the process continues until an absorbing state s_N is reached. This may be written as

$$\tau = s_0, a_0, r_1, s_1, a_1, r_2, \ldots, r_N, s_N$$

where τ is a *trajectory* (or *episode*) and corresponds to a *path* reaching a goal state.

Given a policy θ, we can sample N trajectories and estimate the discounted cumulative reward (defined in Section 1.5.4) as follows:

$$V^\theta(s) = \frac{1}{N} \sum_{i=1}^{N} R(\tau_i)$$

where R is a function that maps τ_i to a reward.

Note again that this allows us to estimate V^θ without knowledge of the underlying MDP. Of course, we are free to choose the sampling procedure and this may in turn affect the accuracy and robustness of our learning – one naive way of doing so is by following a Monte Carlo approach [155, Ch. 5]

The V^θ function may be decomposed using the so-called *Q-function*. The Q-function is commonly used in reinforcement learning. It predicts the expected total reward of an agent taking action a in state s and then following a particular policy θ thereafter. The notation $Q^\theta(s, a)$ refers to the expected cumulative reward the agent will receive starting

from state s, taking action a, and then following policy θ for all future actions. Specifically, it is defined as the expected sum of discounted rewards:

$$Q^\theta(s,a) = E\left(\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid \theta, s_0 = s, a_0 = a\right)$$

where $\gamma \in [0,1]$ is a discount factor that determines the relative importance of immediate versus future rewards, and r_t is the reward the agent receives at time t.

Specifically, the value of being in state s under policy θ is given by the expected value of taking an action a in state s and then following policy θ thereafter, weighted by the probability of taking that action under the policy:

$$V^\theta(s) = \sum_a \theta(s,a) Q^\theta(s,a)$$

Conversely, the Q-function can be computed recursively from the value function V^θ. It expresses the expected Q-value of taking an action a in state s as the expected immediate reward plus the expected discounted value of being in the resulting state s' and following policy θ thereafter:

$$Q^\theta(s,a) = \sum_{s'} T^\theta(s,a,s') \left(R(s,a,s') + \gamma V^\theta(s')\right)$$

where $T^\theta(s,a,s')$ is the probability of transitioning from state s to state s' when taking action a under policy θ, and $R(s,a,s')$ is the expected immediate reward when transitioning from state s to state s' by taking action a.

Similarly to the expected cumulative reward case, we can learn the Q-function and the associated policy according to the Q-Learning algorithm [155, sec. 6.5], which goes as follows:

Algorithm 13: Q-Learning

Data: MDP $= (\mathcal{S}, \mathcal{A}ct, T, \iota, R)$, discounting factor γ
Result: Optimal policy θ

```
Initialize Q arbitrarily;
θ ← ϵ-greedy policy with respect to Q;
foreach episode do
    Initialize s;
    while s is not terminal do
        a ← θ(s);
        Take action a, observe r and the new state s′;
        δ ← r + γ max_{a′} Q(s′, a′) − Q(s, a);
        Q(s, a) ← Q(s, a) + αδ;
        θ ← ϵ-greedy policy with respect to Q;
        s ← s′;
    end
end
θ ← greedy policy with respect to Q
```

where an ϵ-greedy policy with respect to Q is one that picks the "best" action a for a state s with a probability $1-\epsilon$, and picks an action at random with probability ϵ (for a small ϵ). A greedy policy is an ϵ-greedy policy where $\epsilon = 0$. The algorithm initializes the Q-function arbitrarily and uses an epsilon-greedy policy with respect to Q to select actions. Then, it starts iterating over episodes. For each episode, the algorithm initializes the current state s and starts a loop until the state s is terminal. At each step, the algorithm selects an action a based on the current policy θ. It then takes action a, observes the reward r and the new state s', it calculates the temporal difference error δ as the difference between the Q-value of the current state-action pair and the estimated value of the best action in the next state, and updates the Q-function accordingly using δ and a learning rate α. The algorithm then updates the policy based on the updated Q-function and sets the current state to the new state s'. The algorithm repeats this process for multiple episodes until convergence. The final output of the algorithm is the optimal policy θ that maximizes the expected cumulative reward.

Note that Q-Learning is a suitable reinforcement learning technique when the state-action space is small enough to be represented by a Q-table, but for larger and more complex problems, alternative methods must be employed, such as Deep Q-Networks [120] or Actor-Critic [154, Ch. 13].

5.5 Towards Markov Deduction Structures

The basic intuition beyond our framework is that resource-bounded reasoning in real-world agents can be represented as a state space exploration process. The state space we consider is a very peculiar one: it includes states that are finite sets of Boolean formulae representing the information about the world actually available to an agent. We refer to such a peculiar space as the *information space*. A reasoning process consists of an agent that moves within the information space by selecting actions that correspond to rules.

Let i be a generic notation for an agent and let AP^i be the set of atomic propositions that i uses to represent elementary facts of the world. We assume AP^i to be finite. The *knowledge representation language* (KRL) of an agent $\mathcal{L}_{AP^i}$ is the set of all formulae ϕ of *maximum lenght* t obtainable by combining atoms in AP^i through standard Boolean connectives. Here, the lenght of a formula ϕ is intended as the number of *literals* occurring in ϕ and depends on the memory resources of the agent. Each KRL $\mathcal{L}_{AP^i}$ is associated with an information space $\mathcal{S}^i$ that corresponds to its power set, i.e., $\mathcal{S}^i := 2^{\mathcal{L}_{AP^i}}$. Each element $s^i \in \mathcal{S}^i$ is called an *information state* and consists of a finite collection of finite-lenght formulae ϕ representing the information actually available to i. A *rule* a^i is an action that allows i to infer new formulae from those included in its actual information state s^i and thus reach a new information state $s^{i'}$ including all the formulae in s^i plus the formulae inferred from s^i through a^i. All the rules that i knows and can apply are collected in a finite set $\mathcal{A}ct^i$. Analogously to the actions of an MDP, the rules a^i are typically non-deterministic. That is, there may be more than one possible state $s^{i'}$ that can be reached from s^i by one-step application of $a^i \in \mathcal{A}ct$. For example, consider the state $s^i := \{P, P \to Q, P \to R\}$ and the rule $a^i :=$"from ϕ and $\phi \to \psi$, infer ψ". In this scenario, there exist two different possible states that can be reached from s^i by one-step application of a, i.e., $s^{i'} := \{P, P \to Q, P \to R, Q\}$, and $s^{i''} := \{P, P \to Q, P \to R, R\}$. Given the selected rule, the choice between $s^{i'}$ and $s^{i''}$ entirely depends on the probability distribution describing the behaviour of i.

In what follows, we use notation $s^i \to_{a^i} s^{i'}$ to state that $s^{i'}$ can be reached from s^i by one-step application of the rule a^i. If $s^i \to_{a^i} s^{i'}$, then we call $s^{i'}$ an a^i-*successor* of s^i. The set of all a^i-successors of a given state s^i is denoted by $\mathrm{Succ}_{a^i}(s^i)$. Inferences in our framework are represented as sequences of alternating states and rules (actions). By analogy with paths of MDPs, we denote inferences by π, their states by $\pi(t), t \in \mathbf{N}$ and their rules by $\alpha_\pi(t)$. An inference π is said *correct* if and only if, for all $t \geq 0$, it holds that $\pi(t+1) \in \mathrm{Succ}_{\alpha_\pi(t)}(\pi(t))$. The key point is that not all correct inferences are practically feasible for agents with finite computational resources. This is due to the fact that every time an agent applies a rule, it incurs in a computational cost that erodes its resources. In this work, we assume that each rule $a^i \in \mathcal{A}ct^i$ has a fixed computational cost that we denote by $cost(a^i)$. The cumulative cost of a finite inference $\pi := \pi(0), \alpha_\pi(0), \ldots, \pi(t)$

of lenght t, denoted $Cost(\pi)$, is then defined as $\sum_{k=0}^{t-1} cost(\alpha_{\pi(k)})$. Let $resources^i \in \mathbf{N}$ denote the finite amount of computational resources available for an agent i. An inference π is considered *feasible* for i if and only if $Cost(\pi) \leq resources^i$. At the same time, we say that an inference π is *optimal* if and only if: (i) π is feasible, and (ii) for all other feasible inferences π', $Cost(\pi) \leq Cost(\pi')$. We can then introduce a notion of fundamental relevance for our framework, that of *optimal derivability*, which is defined as follows.

Definition 81 (Optimal derivability) *For any given formula ϕ in the KRL $\mathcal{L}_{AP^i}$, we say that ϕ is* optimally derivable *for i if and only if there exists at least one optimal inference π leading from the current information state s^i to an information state $s^{i'}$ that includes ϕ.*

Our final aim is to define a procedure to check whether a formula ϕ is optimally derivable for i given its available resources. To this aim, we introduce a compact representation of i in terms of an algebraic construct that we call *Markov Deduction Structure* (MDS). The name recalls Konolige's deduction models mentioned above, which inspired our framework.

Definition 82 (Markov deduction structure.) *A MDS for an agent i, denoted M^i_{MDS}, is defined as a tuple:*

$$\langle \mathcal{L}_{AP^i}, \mathcal{S}^i, \mathcal{A}ct^i, cost^i, resources^i \rangle$$

where:

- *$\mathcal{L}_{AP^i}$ is a KRL defined over a finite set of atoms AP^i;*
- *$\mathcal{S}^i$ is the information space relative to $\mathcal{L}_{AP^i}$;*
- *$\mathcal{A}ct^i$ is a finite non empty set of rules;*
- *$cost^i$ is a cost vector that assigns a non-negative integer to each $a^i \in \mathcal{A}ct^i$ representing the cost of a^i;*
- *$resources^i \in \mathbf{N}$ is a non-negative integer representing the total amount of computational resources available for an agent i.*

Given a MDS M^i_{MDS} with KRL $\mathcal{L}_{AP^i}$, notation $s^i \vdash_{M^i_{\mathsf{MDS}}} \phi$ is used to denote that formula $\phi \in \mathcal{L}_{AP}$ is optimally derivable in M_{MDS} given initial state s^i.

5.6 Derivability Checking

Once introduced the concept of MDS, the next task is to define a procedure to check whether $s^i \vdash_{M^i_{\mathsf{MDS}}} \phi$ holds in a certain MDS M^i_{MDS}. With a slightly abuse of terminology, we refer to this task as *derivability-checking* and present two different strategies for addressing it. The first strategy, described in Section 5.6.1, requires to specify and solve the MDP *associated* to a given MDS. The second strategy, described in Section 5.6.2, is based on reinforcement learning and requires only the specification of a reward function.

5.6.1 Derivability-Checking with Markov Decision Processes

Given a MDS M^i_{MDS}, we denote its associated MDP by $\mathcal{M}^i_{\mathsf{MDS}}$. The latter is constructed as follows:

- the states space and the set of actions of the decision process $\mathcal{M}^i_{\mathsf{MDS}}$ are the same of the relative Markov deduction structure M^i_{MDS};
- the specification of the transition matrix $T^i : \mathcal{S} \times \mathcal{A}ct \times \mathcal{S}^i \to [0,1]$ is left free to the modeller and depends on how they want to characterize the probabilistic behaviour of i. The only mandatory constraint is that for each $s^i, s^{i'} \in \mathcal{S} \times \mathcal{S}$:

$$T^i(s^i, a^i, s^{i'}) := \begin{cases} \geq 0 & \text{if } s \in \mathrm{Succ}_{a^i}(s^i)\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

  This constraint ensures that i applies the rules correctly. A practical way to simplify the task consists in assuming $T(s^i, a^i, s^{i'})$ to be the same for all $s^{i'} \in Succ_{a^i}(s^i)$. In this case, the transition matrix can be easily generated as follows:

$$T(s^i, a^i, s^{i'}) := \begin{cases} 0 \text{ if } s^{i'} \notin Succ_{a^i}(s^i)\,, \\ \frac{1}{|Succ_{a^i}(s^i)|} \text{ otherwise}\,. \end{cases}$$

- the specification of the reward function $R : \mathcal{S}^i \times \mathcal{A}ct^i \times \mathcal{S}^i \to \mathbf{N}$ is obtained as follows. For each $s^i, a^i, s^{i'} \in \mathcal{S}^i \times \mathcal{A}ct^i \times \mathcal{S}^i$:

$$R(s^i, a^i, s^{i'}) := \begin{cases} -cost(a^i) & \text{if } s^{i'} \in \mathrm{Succ}_a(s^i)\,, \\ -\infty & \text{otherwise}\,. \end{cases}$$

That is, if the agent correctly applies a^i, then it gets a negative reward equals to $-cost(a^i)$, otherwise it incurs in an infinite cost. In practice, this trick ensures that all transitions

stemming from an incorrect application of a rule are blocked and, thus, the agent can follow only correct inferences.

Each policy θ in $\mathcal{M}^i_{\mathsf{MDS}}$ identifies a sequence of rules that allow the agent to derive a desired conclusion ϕ from an initial set of premises Γ. The goal event Φ is represented by the set of states $s^{i'} \in \mathcal{S}^i$ including our desired conclusion ϕ, i.e., $\Phi := \{s^{i'} \in \mathcal{S}^i : \phi \in s^{i'}\}$, while the initial state $s^i \in \mathcal{S}^i$ is the state including all the given premises Γ. V^θ_Φ denotes the *expected cumulative cost* incurred by the agent i until reaching a state $s^{i'} \in \Phi$. That is, V^θ_ϕ represents the expected cumulative cost incurred by the agent to derive ϕ from Γ applying the rules as specified by θ. An optimal policy θ^* is a policy that *minimizes* the expected cumulative cost incurred by i to derive ϕ from Γ. In practice, an optimal policy in this context corresponds to a sequence of rules that allows the agent to derive a certain conclusion from a given set of premises *by making the best use of its computational resources*. Given this definition of optimal policy, we can re-define the concept of *optimal derivability* of a formula $\phi \in \mathcal{L}_{AP^i}$ with respect to the MDS M^i_{MDS} representing the agent i as follows.

Definition 83 (Optimal derivability in MDS) *Let M^i_{MDS} be a MDS and let $\mathcal{M}^i_{\mathsf{MDS}}$ its associated Markov decision process. Given a formula $\phi \in \mathcal{L}_{AP^i}$, we say that $s^i \vdash_{M^i_{\mathsf{MDS}}} \phi$ holds if and only if there exist an optimal policy θ^* in $\mathcal{M}^i_{\mathsf{MDS}}$ such that $V^{\theta^*}_\Phi \leq resources^i$.*

Algorithm 14 provides an automatic procedure to check whether $s^i \vdash_{M^i_{\mathsf{MDS}}} \phi$ holds in a given MDS M^i_{MDS} based on building and solving the associated MDP $\mathcal{M}^i_{\mathsf{MDS}}$. The algorithm works as follows. The first step consists of building the associated MDP $\mathcal{M}^i_{\mathsf{MDS}}$ from M^i_{MDS} following the strategy explained above in Section 5.6.1. Notably, this step involves to specify a probability value for each tuple $(s^i, a^i, s^{i'}) \in \mathcal{S}^i \times \mathcal{A}ct^i \times \mathcal{S}^i$. The modeller can perform this task manually, by selecting a proper probability value for each $(s^i, a^i, s^{i'}) \in \mathcal{S}^i \times \mathcal{A}ct^i \times \mathcal{S}^i$, or automatically by assuming $T(s^i, a^i, s^{i'})$ to be the same for all $s^{i'} \in Succ_{a^i}(s^i)$ and zero for all $s^{i'} \notin Succ_{a^i}$. The latter is the strategy implemented in Algorithm 14. The second step consists of constructing a modified transition matrix $\mathbf{T}$ that is obtained by making all goal states (i.e., all $s^{i'}$ s.t. $\phi \in s^{i'}$) absorbing. Insofar we are interested in learning an optimal policy that reaches a given conclusion ϕ, this trick ensures us that the procedure terminates when a goal state (i.e., a state including our desired conclusion ϕ) is reached. The third step consists of using the associated MDP with the modified transition matrix $\mathbf{T}$ to learn an optimal policy θ^*. This is obtained via dynamic programming as explained in Section 1.5.4. The final step consists of computing $V^{\theta^*}_\Phi$ through Equation (1.25) and checking whether $V^{\theta^*}_\Phi \leq resources^i$: if this is the case, then the algorithm returns "YES", otherwise it returns "NO".

Algorithm 14: Derivability Checking

Data: MDS $= \langle \mathcal{L}_{AP^i}, \mathcal{S}^i, \mathcal{A}ct^i, cost^i, resources^i \rangle$, formula $\phi \in \mathcal{L}_{AP^i}$, initial state $s^i \in \mathcal{S}^i$

Result: "yes" or "no"

build the associated MDP $\mathcal{M}^i_{\mathsf{MDS}}$:

foreach $s^i, a^i, s^{i'} \in \mathcal{S}^i \times \mathcal{A}ct^i \times \mathcal{S}^i$ **do**

$$T(s^i, a^i, s^{i'}) := \begin{cases} 0 \text{ if } s^{i'} \notin Succ_{a^i}(s^i) \\ \frac{1}{|Succ_{a^i}(s^i)|} \text{ otherwise}. \end{cases}$$

end

foreach $s^i, a^i, s^{i'} \in \mathcal{S}^i \times \mathcal{A}ct^i \times \mathcal{S}^i$ **do**

$$R(s^i, a^i, s^{i'}) := \begin{cases} -cost(a^i) \text{ if } s^{i'} \in Succ_{a^i}(s^i) \\ -\infty \text{ otherwise}. \end{cases}$$

end

make goal states absorbing:

foreach $s^{i'} \in \mathcal{S}^i : \phi \in s^{i'}$ **do**

$$T(s^{i'}, a^i, s^{i''}) = \begin{cases} 0 \text{ if } s^{i'} \neq s^{i''} \\ 1 \text{ otherwise}. \end{cases}$$

end

calculate θ^* through Algorithms 2 and 1,

calculate $V^{\theta^*}_{\phi}(s^i)$ through Algorithm 1,

if $V^{\theta^*}_{\phi}(s^i) \leq resources^i$ **then**

return: YES;

else

return: NO;

end

5.6.2 Derivability-Checking with Reinforcement Learning

The strategy outlined in the previous section requires a full specification of the transition matrix T^i characterizing the probabilistic behaviour of the agent. In practice, this is usually difficult to obtain as the dimensionality of the information space $\mathcal{S}^i$ is too big[7]. For this reason, a more feasible strategy we can adopt for derivability checking consists of approximating the optimal policy θ^* by applying model-free RL techniques, such as the Q-learning introduced above in Section 5.4.1. This strategy is implemented in Algorithm

[7]Remember that $\mathcal{S}^i$ is defined as the power set of the KRL $\mathcal{L}_{AP^i}$, which is in turn the set of all finite-length formulae that can be built by combining atoms in AP^i with standard Boolean connectives.

15, which proceeds as follows. First, it builds a specification of the reward function R^i by using the cost vector $Cost^i$ as specified above. Second, it makes absorbing the goal states by setting $R(s^i, a^i, s^{i'})$ for all $s^i \in \Phi$ as follows:

$$R(s^i, a^i, s^{i'}) = \begin{cases} -\infty \text{ if } s^{i'} \neq s^i\,, \\ 0 \text{ otherwise}\,. \end{cases}$$

Third, it calculates the optimal policy θ^* via Q-learning and the modified reward function R. Finally, it checks whether $V^{\theta^*} \leq resources^i$.

Algorithm 15: Model-free Derivability Checking

Data: MDS $= \langle \mathcal{L}_{AP^i}, \mathcal{S}^i, \mathcal{A}ct^i, cost^i, resources^i \rangle$, formula $\phi \in \mathcal{L}_{AP^i}$, initial state $s^i \in \mathcal{S}^i$

Result: "yes" or "no"

build the reward function R:
foreach $s^i, a^i, s^{i'} \in \mathcal{S}^i \times \mathcal{A}ct^i \times \mathcal{S}^i$ **do**
  $R(s^i, a^i, s^{i'}) := \begin{cases} -cost(a^i) \text{ if } s^{i'} \in Succ_{a^i}(s^i) \\ -\infty \text{ otherwise}\,. \end{cases}$
end
make goal states absorbing:
foreach $s^{i'} \in \mathcal{S}^i : \phi \in s^{i'}$ **do**
  $R(s^{i'}, a^i, s^{i''}) = \begin{cases} -\infty \text{ if } s^{i'} \neq s^{i''} \\ 0 \text{ otherwise}\,. \end{cases}$
end
calculate θ^* through Algorithm 13 ,
calculate $V_\phi^{\theta^*}(s^i)$,
if $V_\phi^{\theta^*}(s^i) \leq resources^i$ **then**
  return: YES;
else
  return: NO;
end

5.6.3 Example

To illustrate our framework, consider the Markov Deduction Structure $\langle \mathcal{L}_{AP^i}, \mathcal{S}^i, \mathcal{A}ct^i, cost^i, resources^i \rangle$ such that:

- $AP^i := \{P, R\}$;

- $\mathcal{L}_{AP^i}$ is the set of formulae of length 4 that can be obtained by combining atoms in AP^i through the Boolean connectives $\neg$, $\wedge$, $\vee$, and $\rightarrow$;
- $\mathcal{S}^i := 2^{\mathcal{L}_{AP^i}}$;
- $\mathcal{A}ct^i := \{E\wedge;\ MP; DS\}$, where $E\wedge$ is the usual rule of conjunction elimination, MP is the modus ponens, and DS is the disjunctive syllogism;
- $cost^i$ is defined so that $cost^i(E\wedge) = 0$, $cost^i(MP) = 1$ and $cost^i(DS) = 2$;
- $resources^i = 2$.

Let $s_0^i := \{P \vee R;\ \neg P;\ P \vee R \rightarrow R \wedge R\}$ be the initial state and let R be our desired conclusion. Our task is to check whether $s_0^i \vdash_{M^i_{\mathsf{MDS}}} R$, which corresponds to checking whether there exists a policy θ that allows i to reach a state containing R without exceeding its computational resources.

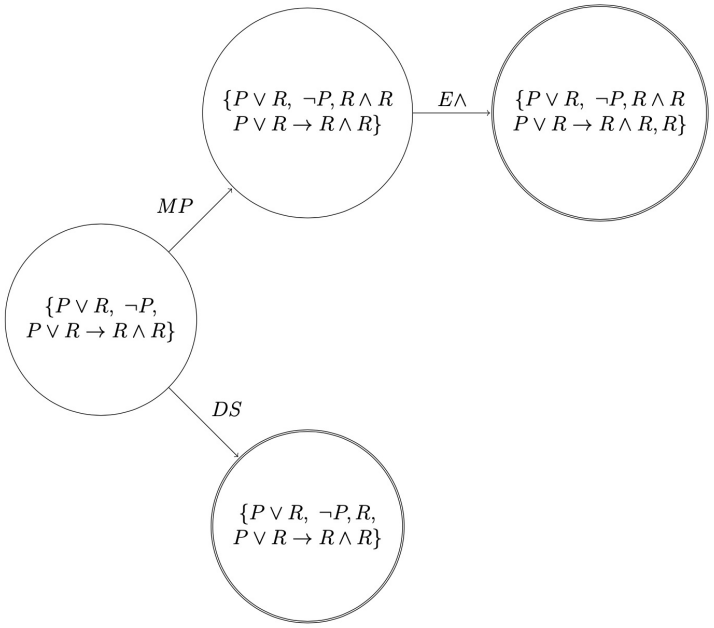


Figure 5.3: Representation of the associated MDP

We can solve the problem by finding the *optimal policy* θ^* that allows the agent to reach the target information state containing R from the initial state, and then check whether its expected cumulative cost does not exceed the agent's available resources, here amounting to 2. In this example, there are two policies leading to an information state that contains R, namely θ_1 and θ_2 such that $\theta_1(s_0^i) = MP$ and $\theta_1(s_1^i) = E\wedge$, where $s_1^i = s_0^i \cup \{Q \wedge Q\}$; and $\theta_2(s_0^i) = DS$.

Since both policies generate only one possible path, we can compute V_R^θ by summing up the cost of the rules, i.e., $V_R^{\theta_1} = 1$ and $V_R^{\theta_2} = 2$. Therefore, θ_1 is optimal. To determine whether $s_0^i \vdash_{M^i_{\mathsf{MDS}}} R$, we simply check if $V_R^{\theta_1} \leq 2$. In fact, $V_R^{\theta_1} = 1$, and we can conclude that proposition R is effectively derivable in the MDS of i.

5.7 The Logic of Markov Deduction Structures

This section introduces a logic of knowledge for real-world agents with finite resources and inferential abilities based on the formalism of Markov deduction structures previously introduced. We call it *Markov Deduction Structures Logic* (MDSL).

5.7.1 Syntax

Let **At** be a countable finite set of atomic propositions p, let $\mathcal{A}$ be a finite non-empty set of agents indexed by $i \in \mathcal{A}$ and let $\Gamma \subseteq \mathcal{A}$ denote a finite non-empty group of agents. The language $\mathcal{L}_{\mathsf{MDSL}}$ of the Markov deduction structures logic is defined as follows:

$$\phi := \quad p \mid \neg\phi \mid \phi_1 \wedge \phi_2$$

$$\epsilon := \quad A^i\phi \mid K^i\phi \mid E^\Gamma\phi \mid D^\Gamma\phi,$$

$$\lambda := \quad \phi \mid \epsilon \mid \neg\lambda \mid \lambda_1 \wedge \lambda_2\,.$$

The language is composed by three kinds of formulae. The ϕ-formulae include atoms p and standard Boolean connectives for negation and disjunction. These formulae are used to represent both facts in the world and the information about the world that an agent may possess. The ϵ-formulae include four different epistemic operators for *awareness* A^i, *knowledge* K^i, *everybody knows* E^Γ, and *distributed knowledge* D^Γ. The epistemic operators nest ϕ-formulae and have the following reading:

- $A^i\phi$: "agent i is aware that ϕ";
- K^i "agent i explicitly knows ϕ";
- $E^\Gamma\phi$ "everybody in Γ explicitly knows ϕ";
- $D^\Gamma\phi$ "it is distributed explicit knowledge in the group of agents Γ that ϕ".

Notice that iterated nested epistemic operators are not allowed. For the same reason, the language does not include an operator for common knowledge. A third meta-variable λ is introduced to denote the combination of ϕ and ϵ-formulae by Boolean connectives.

5.7.2 Semantics

The semantics of the MDSL is obtained by merging the standard possible-worlds semantics for propositional logic with the formalism of Markov deduction structures introduced above. Notably, the satisfiability conditions for ϕ-formulae are defined as in classical

propositional logic, while the satisfiability conditions for ϵ-formulae are based on Markov deduction structures.

Definition 84 (Model (MDSL)) *A model M_{MDSL} in the MDSL is defined as a tuple $\langle \Omega, \mathcal{A}, \{\mathcal{M}^i_{\mathsf{MDS}}\}_{i\in\mathcal{A}}, \{@^i\}_{i\in\mathcal{A}}, V\rangle$, where:*

- *Ω is a set of possible worlds;*
- *$\mathcal{A}$ is a finite non-empty set of agents;*
- *For each $i \in \mathcal{A}$, $\mathcal{M}^i_{\mathsf{MDS}}$ is a MDS that satisfies the condition $AP^i \subseteq \mathbf{At}$;*
- *For each $i \in \mathcal{A}$, $@^i : \Omega \to \mathcal{S}^i$ is a function that assigns to each world $\omega \in \Omega$ an information state $s^i \in \mathcal{S}^i$ including all propositions true in ω that the agent is actually aware of;*
- *$V : \Omega \times \mathcal{L}_{\mathsf{MDSL}}(\mathbf{At}) \to \{0, 1\}$ is a valuation function that assigns a Boolean value to each pair composed by a possible world $\omega \in \Omega$ and an atom $p \in \mathcal{L}_{\mathsf{MDSL}}(\mathbf{At})$.*

For ϕ-formulae, the satisfiability conditions are defined with respect to a model M_{MDSL} and a world $\omega \in \Omega$ as follows:

$$\begin{array}{ll} M_{\mathsf{MDSL}}, \omega \models p & \text{iff } V(\omega, p) = 1\,, \\ M_{\mathsf{MDSL}}, \omega \models \neg\phi & \text{iff } M_{\mathsf{MDSL}}, \omega \not\models \phi\,, \\ M_{\mathsf{MDSL}}, \omega \models \phi_1 \wedge \phi_2 & \text{iff } M_{\mathsf{MDSL}}, \omega \models \phi_1 \text{ and } M_{\mathsf{MDSL}}, \omega \models \phi_2\,, \end{array}$$

For what concerns the awareness operator A^i, its semantics strictly resembles the usual one provided in the *awareness structures* [167, p. 82]. That is, it relies on a awareness function $@^i$ connecting each possible world $\omega \in \Omega$ to an information states $s^i \in \mathcal{S}^i$ including all the information about ω (i.e., all the ϕ-formulae true of ω) that the agent is aware of. The satisfiability condition for A^i is accordingly defined as follows:

$$M_{\mathsf{MDSL}}, \omega \models A^i\phi \quad \text{iff} \quad \phi \in @^i(\omega)\,,$$

where $@^i(\omega)$ denotes the information state $s \in \mathcal{S}^i$ that $@^i$ assigns to ω.

The semantic of the *explicit knowledge* operator K^i is defined in terms of optimal derivability in the MDS representing the agent i:

$$M_{\mathsf{MDSL}}, \omega \models K^i\phi \quad \text{iff} \quad @^i(\omega) \vdash_{M^i_{\mathsf{MDS}}} \phi\,.$$

That is, an agent i *explicitly knows* ϕ if and only if ϕ is optimally derivable in M^i_{MDS} starting from the information the agent is aware of in ω, i.e., $@^i(\omega)$.

For what concerns the *everybody-know* operator, the usual intuition is that the formula ϕ nested within the scope of E^Γ represents information that all agents in Γ know *independently* from any possible mutual interaction or cooperation among them. In other worlds, $E^\Gamma\phi$ is the case if and only if all agents in Γ *individually* know (explicitly) ϕ. In the semantics of MDSL, this intuition is captured simply requiring that ϕ is optimally derivable in all the MDSs M^i_{MDS} of all agents $i \in \Gamma$. The satisfiability condition for the *everybody knows* operator is accordingly defined as follows:

$$M_{\mathsf{MDSL}}, \omega \models E^\Gamma\phi \quad \text{iff} \quad \forall i \in \Gamma : \ @^i(\omega) \vdash_{M^i_{\mathsf{MDS}}} \phi\,.$$

For the *distributed knowledge* operator, the usual intuition is that the nested formula ϕ represents information that all the agents in Γ explicitly know *given that they cooperate each other and mutually share the information they possess.* In our framework, we assume that agents cooperate whenever they share the following three fundamental elements:

1. the propositions they actually are aware of;

2. the rules of inference they know;

3. the computational resources they possess.

We model cooperation among agents in Γ by introducing a *joint* MDS $\mathcal{M}^\Gamma_{\mathsf{MDS}}$ and a *joint awareness function* $@^\Gamma$, which are defined as follows.

Definition 85 (Joint Markov deduction structure) *Given a group of agents $\Gamma \subseteq \mathcal{A}$ and their respective MDSs $M^i_{\mathsf{MDS}}, i \in \Gamma$, the* joint *MDS M^Γ_{MDS} is defined as a tuple $\langle \mathcal{L}_{AP^\Gamma}, \mathcal{S}^\Gamma, \mathcal{A}ct^\Gamma, cost^\Gamma, resources^\Gamma \rangle$, where:*

- *$\mathcal{L}_{AP^\Gamma}$ is a joint KRL defined over the finite set of atoms AP^Γ obtained as $\bigcup_{i\in\Gamma} AP^i$;*
- *$\mathcal{S}^\Gamma$ is a joint information space obtained as 2^{AP^Γ};*
- *$\mathcal{A}ct^\Gamma$ is a joint set of rules obtained as $\bigcup_{i\in\Gamma} \mathcal{A}ct^\Gamma$;*
- *$cost^\Gamma$ is the joint cost vector such that, for each $a \in \mathcal{A}ct^\Gamma$, $cost^\Gamma(a) := \min_{i\in\Gamma} cost^i(a)$*[8]
- *$resources^\Gamma$ is the joint amount of resources obtained as $\sum_{i\in\Gamma} resources^i$.*

[8]Here we assume that, in a cooperation scenario, for each rule $a \in \mathcal{A}$, the agents in Γ collectively decide that the agent in charge of applying the rule a is the one among them incurring in the least cost for the application of that given.

Definition 86 (Joint awareness function) *Given a group of agents* $\Gamma \subseteq \mathcal{A}$, the joint awareness function $@^{\Gamma}$ *is defined as a function that assigns to each* $\omega \in \Omega$ *the information state* $s^{\Gamma} \in \mathcal{S}^{\Gamma}$ *corresponding to the union of all information states of the various* $i \in \mathcal{A}$, *i.e.,*

$$(\forall \omega \in \Omega)\ @^{\Gamma}(\omega) := \bigcup_{i \in \Gamma} @^{i}(\omega)\,.$$

Given both $\mathcal{M}^{\Gamma}_{\mathsf{MDS}}$ and $@^{\Gamma}$, the satisfiability condition for the *distributed knowledge* operator is accordingly defined as follows:

$$M_{\mathsf{MDSDL}}, \omega \models D^{\Gamma}\phi \quad \text{iff} \quad @^{\Gamma}(\omega) \vdash_{M^{\Gamma}_{\mathsf{MDS}}} \phi\,.$$

5.7.3 MDSL and Logical Omniscience

The MDSL framework provides a satisfactory solution to the problem of logical omniscience as it satisfies all the requirements that we have above listed:

1. It solves all the relevant forms of logical omniscience, namely: closure under logical consequence (and its special cases) and closure under tautologies;

2. It represents agents as moderately competent;

3. It represents agents as knowing all and only the formulae they can infer given their inferential abilities and making the best use of their resources.

To prove (i), it is sufficient to observe that both models in which closure under logical consequence does not hold and scenarios in which closure under tautologies does not hold can be coherently defined in MDSL semantics. For the former, consider models in which $\{\omega : v(\omega, \phi_1) = 1\} \subseteq \{\omega : v(\omega, \phi_2) = 1\}$ and $@^{i}(\omega) \vdash_{M^{i}_{\mathsf{MDS}}} \phi_1$ hold but $@^{i}(\omega) \vdash_{M^{i}_{\mathsf{MDS}}} \phi_2$ does not hold. For the latter, consider models in which $\{\omega : v(\omega, \phi) = 1\} = \Omega$ but $@^{i}(\omega) \nvdash_{M^{i}_{\mathsf{MDS}}} \phi$. The same reasoning holds for the multi-agent versions of explicit knowledge.

For what concerns (ii), simply observe that logically ignorance is avoided whenever at least one non-trivial rule is included in the set of rules $\mathcal{A}ct^{i}$.

Finally, notice that the semantics of the explicit knowledge operator K^{i} and its multi-agent counterparts (E^{Γ} and D^{Γ}) are based on the notion of *optimally derivability* $\vdash_{M^{i}_{\mathsf{MDS}}}$ introduced in Definition 83, which captures the requirement expressed by (iii). For what concerns limitations given by resources, these are modelled by considering feasible only the inferences whose cumulative cost does not exceed the resources the agent has available. For what concerns the clause related to "making the best use of the resources", this is implemented in the semantics by connecting the notion of optimal derivability (and, thus,

explicit knowledge) to the optimal policy θ^*, which represents the sequence of actions that allows the agent to reach is inferential goal by minimising the cost incurred and, thus, by making the best possible use of its limited resources.

5.8 The Dynamic Logic of Markov Deduction Structures

The previous section introduced a static framework to reason about knowledge in real-world agents provided with finite amounts of computational resources. It is also of our interest to reason about scenarios in which agents can share or exchange their knowledge, abilities and resources. To this aim, we propose an extension of MDSL with dynamic operators to model exchange of information, inferential know-how and resources between agents. We call this extension *Markov Deduction Structures Dynamic Logic* (MDSDL). This section is dedicated to present its language and semantics.

5.8.1 Syntax

The syntax of MDSDL is defined as follows:

$$\begin{aligned}
\phi &:= \quad p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \\
\epsilon &:= \quad A^i\phi \mid K^i\phi \mid E^\Gamma\phi \mid D^\Gamma\phi\,, \\
\delta &:= \quad \langle a : j\rangle\epsilon \mid \langle r : j\rangle\epsilon \mid \langle[\phi] : j\rangle\epsilon\,, \\
\lambda &:= \quad \phi \mid \epsilon \mid \delta \mid \neg\lambda \mid \lambda_1 \wedge \lambda_2\,.
\end{aligned}$$

The language is obtained by extending the language of MDSL with a dynamic fragment δ including the following dynamic operators:

- $\langle a : j\rangle\epsilon$ is the *cooperative learning* operator representing an agent j that teaches a new rule to another agent i, respectively, a group of agents Γ. It is read:"Given that i (resp. Γ) learns rule a from j, then ϵ holds";

- $\langle r : j\rangle\epsilon$ is the *resources exchange* operator representing an agent j that shares part of its resources with another agent i, respectively, groups of agents Γ. It is read: "Given that j shares resources equals to r with i (resp. Γ), then ϵ holds";

- $\langle[\phi] : j\rangle\epsilon$ is the *information exchange* operator representing an agent j that shares a certain information ϕ with another agent j, respectively, a group of agents Γ. It is read: "Given that j shares information that ϕ with i (resp. Γ), then ϵ holds";

Finally, as for the MDSL, all MDSDL formulae are closed under Boolean negation and conjunction.

5.8.2 Semantics

At the semantic level, dynamic operators correspond to *transformations* that modify the models according to the scenario specified by the operator. In what follows, we introduce a specific transformation for each one of the dynamic operators introduced in the MDSDL language. Based on the latter, we then introduce proper satisfiability conditions for the different kinds of δ-formulae.

Cooperative Learning

Cooperative learning scenarios are characterized by an agent $i \in \mathcal{A}$ (the "student") that learns a new rule a from another agent $j \in \mathcal{A}$ (the "teacher"). Semantically, cooperative learning is modelled by applying a transformation $\tau^{\langle a:j\rangle} : M_{\mathsf{MDS}} \rightarrow M^{\langle a:j\rangle}_{\mathsf{MDSL}}$ mapping the original model M_{MDS} into a transformed model $M^{\langle a:j\rangle}_{\mathsf{MDSL}}$ defined as follows:

- If $a \notin \mathcal{A}ct^j$, i.e., if the teacher does not know the rule a carried by the cooperative learning operator, then $M^{\langle a:j\rangle}_{\mathsf{MDSL}} = M_{\mathsf{MDSL}}$ (the intuition is that if the teacher does not know the rule to be taught, then no learning action actually occurs);

- If $a \in \mathcal{A}ct^j$, then $M^{\langle a:j\rangle}_{\mathsf{MDSL}}$ is obtained by transforming the MDS M^i_{MDS} of agent i into a new MDS $M^{i\,\langle a:j\rangle}_{\mathsf{MDS}}$ such that $\mathcal{A}ct^{i\,\langle a:j\rangle} = \mathcal{A}ct^i \cup \{a\}$, and $cost^{i\,\langle a:j\rangle}$ is obtained by extending $cost^i$ with a new cell including the cost of a (it is not required that $cost^{i\,\langle a:j\rangle}(a) = cost^j(a)$).

A variation of cooperative learning consists of scenarios in which the student consists of a group of agents Γ. Syntactically, these scenarios are modelled by nesting the cooperative learning operator $\langle a : j\rangle$ either over an *everybody-knows* $E^\Gamma\phi$ or a *distributed knowledge* $D^\Gamma\phi$ formula. Semantically, they are modelled as follows. In the case of a cooperative learning operator nesting an *everybody-knows* formula (i.e., $\delta = \langle a : j\rangle E^\Gamma\phi$), the above described transformation $\tau^{\langle a:j\rangle}$ is applied to the MDSs M^i_{MDS} of all agents $i \in \Gamma$. When $j \in \Gamma$, we simple add rule a to the sets of rules of all $i \in \Gamma$ different from j. In the case of a cooperative learning operator nesting a *distributed knowledge* formula (i.e., $\delta = \langle a : j\rangle D^\Gamma\phi$), the above transformation is applied to the joint MDS M^Γ_{MDS}, hence obtaining a

transformed joint MDS $M^{\Gamma\langle a:j\rangle}_{\mathsf{MDS}}$[9]. Notice that, if the cooperative learning operator $\langle a:j\rangle$ nests a distributed knowledge formula $D^{\Gamma}\phi$ and $j \in \Gamma$, then $\langle a:j\rangle D^{\Gamma}\phi$ and $D^{\Gamma}\phi$ are in fact equivalent. Since a is included in $\mathcal{A}ct^j$ and $j \in \Gamma$, then a results to be included both in $M^{\Gamma}_{\mathsf{MDS}}$ and $M^{\Gamma\langle a:j\rangle}_{\mathsf{MDS}}$. In such cases, the cooperative learning operator does not induce any relevant transformation of the model.

Once introduced the notion of transformed model $M^{\langle a:j\rangle}_{\mathsf{MDSL}}$, the satisfiability condition for cooperative learning formulae can be accordingly defined as follows:

$$M_{\mathsf{MDSL}}, \omega \models \langle a:j\rangle\epsilon \quad \text{iff} \quad M^{\langle a:j\rangle}_{\mathsf{MDSL}}, \omega \models \epsilon\,.$$

Resources Exchange.

Resources exchange scenarios are characterized by an agent j (the "exchanger") sharing some of its resources with another agent i, respectively, a group of agents Γ (the "beneficiary"). Semantically, these scenarios are modelled by introducing a transformation $\tau^{\langle r:j\rangle} : M_{\mathsf{MDSL}} \to M^{\langle r:j\rangle}_{\mathsf{MDSL}}$ mapping the original model M_{MDSL} into a transformed model $M^{\langle r:j\rangle}_{\mathsf{MDSL}}$. The latter is obtained by increasing $resources^i$ and decreasing $resources^j$, in the respective MDSs, of the same value r. Notice that the transformation $\tau^{\langle r:j\rangle}$ applies only under the condition that $r \leq resources^j$, i.e., an agent cannot exchange more resources than it has available. If $r > resources^j$, then no transformation occurs and the dynamic formulae is considered not satisfied by default.

Special scenarios of resources exchange are those involving a group of agent Γ as the intended beneficiary of the exchanged resources. Syntactically, these scenarios are modelled by nesting the resource-exchange operator over an *everybody knows* or a *distributed knowledge* formula. Semantically, they are modelled as follows. In the case of a resources exchange operator nesting an everybody-knows formula (i.e., $\delta = \langle r:j\rangle E^{\Gamma}\phi$), the additional resources r are distributed over all agents in Γ. That is, for each $i \in \Gamma$, the transformed MDSs $M^{i\langle r:j\rangle}_{\mathsf{MDS}}$ is obtained by increasing $resources^i$ of a value equals to $\frac{r}{|\Gamma|}$. In the case of a resources exchange operator nesting a distributed knowledge formula (i.e., $\delta = \langle r:j\rangle D^{\Gamma}\phi$), the additional resources r are added directly to $resources^{\Gamma}$ in the joint MDS $M^{\Gamma}_{\mathsf{MDS}}$, hence obtaining a transformed MDS $M^{\Gamma\langle r:j\rangle}_{\mathsf{MDS}}$.

Once introduced the notion of transformed model $M^{\langle r:j\rangle}_{\mathsf{MDSL}}$, the satisfiability condition for the resources exchange operator is accordingly defined as follows:

$$M_{\mathsf{MDSL}}, \omega \models \langle r:j\rangle\epsilon \quad \text{iff} \quad r \leq resources^j \;\wedge\; M^{\langle r:j\rangle}_{\mathsf{MDSL}}, \omega \models \epsilon$$

[9]Notice that the same result can be obtained by adding the rule a to one random $i \in \Gamma$ and thus derive the joint MDS. From how the joint MDS is defined, it follows that it will still include the new a rule.

Information Exchange.

Information exchange scenarios are characterized by an agent j (the "sender") exchanging a piece of information $[\phi]$ with another agent i (the "receiver"). Semantically, they are modelled by introducing a transformation $\tau^{\langle[\phi]:j\rangle} : M_{\mathsf{MDSL}} \to M_{\mathsf{MDSL}}^{\langle[\phi]:j\rangle}$ mapping the original model M_{MDSL} into a transformed model $M_{\mathsf{MDSL}}^{\langle[\phi]:j\rangle}$. The latter is obtained as follows:

- If $\phi \notin @^j(\omega)$, then $M_{\mathsf{MDSL}} = M_{\mathsf{MDSL}}^{\langle[\phi]:j\rangle}$ (the intuition is that if the sender is not aware of the information to be exchanged, then no exchange of information actually occurs);

- If $\phi \in @^j(\omega)$, then the transformed model $M_{\mathsf{MDSL}}^{\langle[\phi]:j\rangle}$ is obtained by transforming the awareness function $@^i$ into a new awareness function $@^{i\,\langle[\phi]:j\rangle}$ such that $@^{i\,\langle[\phi]:j\rangle}(\omega) := @^i(\omega) \cup \{\phi\}$ and, for all $\omega' \neq \omega$, $@^{i\,\langle[\phi]:j\rangle}(\omega) := @^i(\omega)$. In practice, the transformation corresponds to change the actual information state of the receiver passing from $@^i(\omega) \in \mathcal{S}'$ to a new actual information state $@^{i\,\langle[\phi]:j\rangle}(\omega) \in \mathcal{S}^i = @^i(\omega) \cup \{\phi\}$.

Also for information exchange, there are special scenarios involving a group of agent Γ as the intended receiver. Syntactically, these scenarios are modelled by nesting the information exchange operator over an *everybody knows* or a *distributed knowledge* formula. Semantically, they are modelled as follows. In case of an information exchange operator nesting an everybody-knows formula (i.e., $\delta = \langle[\phi] : j\rangle E^\Gamma \phi$), the transformation $\tau^{\langle[\phi]:j\rangle}$ is applied to all $i \in \Gamma$. In case of an information exchange nesting a distributed knowledge formula (i.e., $\delta = \langle[\phi] : j\rangle D^\Gamma \phi$), the transformation $\tau^{\langle[\phi]:j\rangle}$ replaces the original *joint actual information state* $@^\Gamma(\omega)$ with a transformed joint actual information state $@^{\Gamma,\langle[\phi]:j\rangle} := @^\Gamma(\omega) \cup \{\phi\}$.

Once introduced the notion of transformed model $M_{\mathsf{MDSL}}^{\langle[\phi]:j\rangle}$, the satisfiability condition for the information exchange operator is then defined as follows:

$$M_{\mathsf{MDSL}}, \omega \models \langle[\phi] : j\rangle\epsilon \quad \text{iff} \quad M_{\mathsf{MDSL}}^{\langle[\phi]:j\rangle}, \omega \models \epsilon\,.$$

5.9 Model Checking

This section introduces a model checking framework for both MDSL and its dynamic extension. Given a model M_{MDSL}, a world ω and a formula λ, our task consists of finding a procedure that checks whether $M_{\mathsf{MDSL}}, \omega \models \lambda$ holds. For propositional formulae ϕ, the procedure is trivial. It consists of decomposing ϕ in its atomic components, determining their truth-values in ω through V, and finally calculating the truth-value of ϕ by composition using the truth-tables of Boolean connectives. For epistemic and dynamic formulae, we

need to introduce specific procedures based on the derivability-checking task for Markov deduction structures presented in Section 5.5.

5.9.1 MDSL Model Checking

The model checking for MDSL requires to introduce a specific algorithm to check whether a model M_{MDSL} and a world ω satisfy a given MDSL epistemic formula ϵ. This algorithm is reported in Figure 5.4 and works as follows. It starts by taking in input a model M_{MDSL}, a world $\omega \in \Omega$ and a formula ϵ. Depending on the specification of ϵ, the algorithm switches on a proper sub-routine:

1. If $\epsilon = A^i\phi$, the algorithm simply checks whether $\phi \in @^i(\omega)$. If so, it returns "YES"; otherwise, it returns "NO".
2. If $\epsilon = K^i\phi$, the algorithm computes the derivability-checking task $@^i(\omega) \vdash_{M^i_{\mathsf{MDS}}} \phi$ using one of the two algorithms presented in Section 5.6. If the derivability-checking task is satisfied, the algorithm returns "YES"; otherwise, it returns "NO".
3. If $\epsilon = E^\Gamma\phi$, the algorithm computes the derivability-checking task $@^i(\omega) \vdash_{M^i_{\mathsf{MDS}}} \phi$ for each $i \in \Gamma$. If the derivability-checking task is satisfied for all $i \in \Gamma$, then the algorithm returns "YES"; otherwise, it returns "NO".
4. If $\epsilon = D^\Gamma\phi$, the algorithm builds the joint MDS M^Γ_{MDS} and the joint awareness function $@^\Gamma$ as specified in Section 5.7.2. Then, it computes the derivability-checking task $@^\Gamma(\omega) \vdash_{M^i_{\mathsf{MDS}}} \phi$. If the latter is satisfied, the algorithm returns "YES"; otherwise, it returns "NO".

5.9.2 MDSDL Model Checking

The model checking for the MDSLS extends that of MDSL with a proper algorithm to check whether a model M_{MDSL} and a world $\omega \in \Omega$ satisfy a given dynamic formula δ. The algorithm proceeds by applying a transformation that maps the original model M_{MDSL} into a modified model M'_{MDSL}. As specified in Section 5.8.2, the kind of transformation applied depends on the dynamic operator involved in the specified formula δ. Once obtained M'_{MDSL}, the algorithm checks whether $M'_{\mathsf{MDSL}}, \omega$ satisfy the formula ϵ specified within the scope of the dynamic operator by applying Algorithm 5.4. The specific steps of the algorithm are detailed in Figure 5.5.

5.10 Conclusions and Further Works

In this chapter, we have proposed a novel framework for reasoning about real-world agents' knowledge based on representing agents' reasoning as a state-space exploration process

modelable via Markov decision processes and Reinforcement learning. Our approach provides an effective solution to the problem of logical omniscience insofar agents are represented as knowing *all and only the logical consequences of the information they are aware of making the best use of their limited computational resources and inferential abilities*. Compared to other solutions proposed so far for the problem of logical omniscience, ours qualifies as innovative to the extent that it is the first to explicitly consider *learning* as a fundamental constitutive ability of resource-bounded reasoning. However, there remain some issues and limitations of our framework that we aim to solve in future works.

On the theoretical side, one major limitation concerns the possibility of expressing nested epistemic modalities, which are not allowed neither in MDSL nor in its dynamic extension. A possible solution for this limitation has been already proposed in [95] for the logic of deduction models and consists of introducing a secondary MDS for each agent $i \in \mathcal{A}$ representing i's *point of view* on its own and other agents' knowledge. However, we believe to be very difficult to implement this solution in the formalism of MDSs, notably as it would entail a proliferation of MDSs that risks to make the models' specification a very hard task. For this reason, we plan to explore other potential solutions for this issue.

On the implementation side, a major problem is represented by the dimension of the state space $\mathcal{S}$, which tends to be very high even when AP includes only a few propositional atoms and a maximum length for the KRL formulae is fixed to a small number of literals per formula. Large state spaces are notoriously difficult to handle efficiently using classical reinforcement learning methodologies, including the *Q-learning* mentioned above. For an efficient implementation of the formalism proposed here, it will therefore be necessary either to explore reinforcement learning methodologies specific for large state spaces, such as *deep Q-learning* (see e.g. [120]), or to identify a strategy to reduce the dimensionality of $\mathcal{S}$ without affecting the expressiveness of the formalism. We demand the resolution of these problems to a future work specifically dedicated to study the implementation and to perform the experimental evaluation of the framework proposed here.

Algorithm 6: Model checking for ϵ-formulae

Data: A model M_{MDSL}, a world $\omega \in \Omega$ and a MDSL formula ϵ
Result: "yes" or "no"

switch ϵ **do**
  case $\epsilon = A^i\phi$ **do**
    if $\phi \in @^i(\omega)$ **then**
      return: "YES"
    end
    else
      return: "NO"
    end
  end
  case $\epsilon = K^i\phi$ **do**
    Check whether $@^i(\omega) \vdash_{M^i_{\mathsf{MDS}}} \phi$ via Algorithm 4 or 5,
    if $@^i(\omega) \vdash_{M^i_{\mathsf{MDS}}} \phi$ **then**
      return: "YES"
    end
    else
      return: "NO"
    end
  end
  case $\epsilon = E^\Gamma\phi$ **do**
    foreach $i \in \Gamma$ **do**
      Check whether $@^i(\omega) \vdash_{M^i_{\mathsf{MDS}}} \phi$ via Algorithm 4 or 5
    end
    if $\forall i \in \Gamma : @^i(\omega) \vdash_{M^i_{\mathsf{MDS}}} \phi$ **then**
      return: "YES"
    end
    else
      return: "NO"
    end
  end
  case $\epsilon = D^\Gamma\phi$ **do**
    Build M^Γ_{MDS},
    Build $@^\Gamma$,
    Check whether $@^\Gamma(\omega) \vdash_{M^\Gamma_{\mathsf{MDS}}} \phi$ via Algorithm 4 or 5,
    if $@^\Gamma(\omega) \vdash_{M^\Gamma_{\mathsf{MDS}}} \phi$ **then**
      return: "YES"
    end
    else
      return: "NO"
    end
  end
end

Figure 5.4: Model Checking Algorithm for MDSL/MDSDL ϵ-formulae

Algorithm 7: Model checking for δ-formulae

Data: A model M_{MDSL}, a world $\omega \in \Omega$ and a MDSDL formula δ
Result: "yes" or "no"

switch δ **do**
  case $\delta = \langle a : j \rangle \epsilon$ **do**
    build $M_{\mathsf{MDSL}}^{\langle a:j \rangle}$:
    if $a \in \mathcal{A}ct^j$ **then**
      $\mathcal{A}ct^{i\,\langle a:j \rangle} := \mathcal{A}ct^i \cup \{a\}$
    end
    else
      $\mathcal{A}ct^{i\,\langle a:j \rangle} := \mathcal{A}ct^i$
    end
  end
  case $\delta = \langle r : j \rangle$ **do**
    if $r \leq resources^j$ **then**
      build $M_{\mathsf{MDSL}}^{\langle r:j \rangle}$:
      $resources^{i\,\langle r:j \rangle} := resources^i + r$
      $resources^{j\,\langle r:j \rangle} := resources^j - r$,
    end
    else
      answer: "NO"
    end
  end
  case $\delta = \langle [\phi] : j \rangle$ **do**
    build $M_{\mathsf{MDSL}}^{\langle [\phi]:j \rangle}$:
    $ⓐ^{i\,\langle [\phi]:j \rangle}(\omega) := ⓐ^i(\omega) \cup \{\phi\}$,
  end
end
Check whether $M_{\mathsf{MDSL}}^{\langle a:j \rangle}, \omega \models \epsilon \mid M_{\mathsf{MDSL}}^{\langle r:j \rangle} \models \epsilon \mid M_{\mathsf{MDSL}}^{\langle [\phi]:j \rangle} \models \epsilon$ via Algorithm 6
if $M_{\mathsf{MDSL}}^{\langle a:j \rangle}, \omega \models \epsilon \mid M_{\mathsf{MDSL}}^{\langle r:j \rangle} \models \epsilon \mid M_{\mathsf{MDSL}}^{\langle [\phi]:j \rangle} \models \epsilon$ **then**
  answer: "YES"
end
else
  answer: "NO"
end

Figure 5.5: Model Checking Algorithm for MDSDL δ-formulae

Chapter 6

Conclusions

Probabilistic model checking nowadays represents a research field of major interest in the context of formal verification. A growing number of papers is published every year in the area, covering both theoretical and methodological topics and implementation aspects. Within its limitations, this thesis aimed at exploring some developments and applications that, although "unconventional", may nonetheless have interesting philosophical and practical implications.

Chapter 3 opened a new line of research at the intersection of model checking and explainable AI that might prove particularly fruitful in the years to come. Indeed, despite the increasing interest in explainable AI, a shared conceptual framework for this research program is still lacking. In particular, there seems to be no agreement among scholars on what in XAI counts as a *good explanation*, as well as on what the desirable properties of a good explanation are. Limiting our analysis to *post-hoc* explanation methods based on surrogate models, in this dissertation we have identified four desirable properties for XAI explanations worth analysing: *transparency*, *accuracy*, *trustworhtiness*, and *explanatory power*. We have provided a formal characterization for all of them and developed a model checking framework to verify whether they are satisfied by our XAI tools. However, much remains to be done in this field. For example, our framework focuses on only one specific dimension of the opacity problem, i.e., *access-opacity*. Different versions, or extensions, of the framework proposed here could be developed in the future to manage other forms of opacity mentioned in the first part of the chapter. Among them, we plan to focus in particular on *causal opacity*, which nowadays represents a form of opacity of major interest for the AI community (see, [126]). Furthermore, the analysis we have proposed is specific for higher levels of abstraction (notably, functional and design specification level), different properties and related formalisms could be developed for lower levels of abstractions and their related issues (e.g. resource allocation).

Chapter 4 addressed a major issue of contemporary probabilistic model checking de-

riving from the *assumption of stationarity* connected with the use of Markov models. This assumption has important limiting consequences on the application potential of model checking techniques, in particular for what concerns the analysis of real-world systems like biological networks. The theory of imprecise probabilities and the related imprecise Markov models offer a natural way out from the problem of stationarity, notably insofar they allow for modelling non-stationary systems (with a sufficient degree of accuracy) while avoiding complexity issues that affect other approaches. A first systematic approach to model checking with imprecise probabilities is advanced in the chapter, including discrete-time Markov chains, Markov reward models, and probabilistic interpreted systems. Nonetheless, many issues still remain to be explored. Several extensions of this framework to continuous-time models could be considered, starting from *imprecise continuous-time Markov chains* [96] that are so relevant for the analysis of many real-world systems. Secondly, the implementation aspects remain to be developed. In particular, we need to understand whether existing model-checkers (like PRISM [100]) can be suitably extended to deal with imprecise models, or whether the latter require the development of dedicated software tools. Finally, a central topic to be explored concerns the application potential of the proposed framework. In this regard, a promising field seems to be the analysis of metabolic networks. In this area, there already exist several attempts to develop probabilistic model checking frameworks that do not require the stationarity assumption. To the best of our knowledge, however, none of them make use of imprecise Markov models and present the same advantages in terms of computational efficiency.

Chapter 5 focused on a well-known and widely discussed issue related to epistemic logic: the *problem of logical omniscience*. An innovative approach to this "old" problem has been presented based on representing real-world agents' reasoning as a space exploration task that can be modelled through Markov decision processes and reinforcement learning. This approach allows us to consider an aspect of resource-bounded reasoning which is usually neglected in all the other approaches to the problem of logical omniscience: *learning*. The assumption is that the rationality of a real-world agent (as opposed to that of an *ideal* agent) lies not only in its being consistent with its basic knowledge, but also in its being able to make the best use of the limited resources it has available. In this regard, a real-world *rational* agent should be represented as neither omniscient nor ignorant, but as knowing all and only the information it can infer making the best use of its resources. Following this idea, agents in our framework are represented as able to learn how to optimize the use of their resources in a deductive inferential task. Consistently with the idea of representing reasoning as a space exploration task, the general framework we adopt is that of reinforcement learning. Among the various advantages of our framework, it's worth mentioning that it creates a bridge between the logical-deductive representation of agents characteristic of epistemic logic (and logic-oriented AI), and the representation of agents as "heuristic reasoners capable of learning" typical of contemporary AI and ma-

chine learning. Also for this line of research, many issues remain to be explored. These include the development of an appropriate language capable of supporting nested epistemic modalities and their related semantics, as well as the resolution of implementation problems concerning the dimensionality of the considered state spaces and their computational tractability. Finally, we are inclined to believe that several fruitful connections could be explored between the framework introduced here and the various approaches to resource-bounded reasoning recently proposed both in the context of epistemic logic and probabilistic model checking.

In conclusion, our hope is that, by showing the theoretical and methodological versatility of probabilistic model checking and Markov models semantics, this dissertation will stimulate the reader to explore developments and applications of these formalisms that transcend the traditional scope of program verification and embrace diverse research areas, both of practical and philosophical interest.

Bibliography

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[2] Thomas Ågotnes. *A logic of finite syntactic epistemic states*. University of Bergen, Department of Informatics, 2004.

[3] Ethem Alpaydin. *Machine Learning, Revised And Updated Edition*. MIT Press, 2021.

[4] Beena Ammanath. *Trustworthy AI: a business guide for navigating trust and ethics in AI*. John Wiley & Sons, 2022.

[5] Robert Andrews, Joachim Diederich, and Alan B Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, 8(6):373–389, 1995.

[6] Nicola Angius, Giuseppe Primiero, and Raymond Turner. The Philosophy of Computer Science. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2021 edition, 2021.

[7] A. Antonucci, D. Huber, M. Zaffalon, P. Luginbühl, I. Chapman, and R. Ladouceur. Credo: A military decision-support system based on credal networks. In *Proceedings of the 16th Conference on Information Fusion (FUSION 2013)*, Istanbul, Turkey, 2013.

[8] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[9] C. Baier, C. Hensel, L. Hutschenreiter, S. Junges, J.P. Katoen, and J. Klein. Parametric markov chains: PCTL complexity and fraction-free gaussian elimination. *Information and Computation*, 272:104504, 2020.

[10] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[11] Pierre Baldi. *Deep Learning in Science*. Cambridge University Press, 2021.

[12] William Bechtel and Robert C Richardson. *Discovering complexity: Decomposition and localization as strategies in scientific research*. MIT press, 2010.

[13] Claus Beisbart. Opacity thought through: on the intransparency of computer simulations. *Synthese*, 199(3):11643–11666, 2021.

[14] Nikola Benes, Lubos Brim, Samuel Pastva, and David Safránek. Model checking approach to the analysis of biological systems. In Pietro Liò and Paolo Zuliani, editors, *Automated Reasoning for Systems Biology and Medicine*, volume 30 of *Computational Biology*, pages 3–35. Springer, Cham, 2019.

[15] Jamal Bentahar, Bernard Moulin, and John-Jules Ch. Meyer. A new model checking approach for verifying agent communication protocols. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering, CCECE 2006, May 7-10, 2006, Ottawa Congress Centre, Ottawa, Canada*, pages 1586–1590, Ottawa, 2006. IEEE.

[16] Béatrice Bérard, Michel Bidoit, Alain Finkel, François Laroussinie, Antoine Petit, Laure Petrucci, Philippe Schnoebelen, and Pierre McKenzie. *Systems and Software Verification, Model-Checking Techniques and Tools*. Springer, Cham, 2001.

[17] Jakob Bernoulli. *Ars conjectandi: opus posthumum: accedit Tractatus de seriebus infinitis; et Epistola gallice scripta de ludo pilae reticularis*. Impensis Thurnisiorum, 1713.

[18] Francesco Berto and Mark Jago. *Impossible worlds*. Oxford University Press, 2019.

[19] Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007.

[20] Per Bjesse. What is formal verification? *ACM SIGDA Newsletter*, 35(24):1–es, 2005.

[21] Florian J Boge. Two dimensions of opacity and the deep learning predicament. *Minds and Machines*, 32(1):43–75, 2022.

[22] Lubos Brim, Milan Ceska, and David Safránek. Model checking of biological systems. In Marco Bernardo, Erik P. de Vink, Alessandra Di Pierro, and Herbert Wiklicky, editors, *Formal Methods for Dynamical Systems - 13th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2013, Bertinoro, Italy, June 17-22, 2013. Advanced Lectures*, volume 7938 of *Lecture Notes in Computer Science*, pages 63–112, Cham, 2013. Springer.

[23] Jenna Burrell. How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data & Society*, 3(1):2053951715622512, 2016.

[24] Luca Cardelli, Radu Grosu, Kim G Larsen, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin. Lumpability for uncertain continuous-time markov chains. In *International Conference on Quantitative Evaluation of Systems*, pages 391–409, Cham, 2021. Springer.

[25] Rudolf Carnap. *An introduction to the philosophy of science*. Courier Corporation, 2012.

[26] Rudolf Carnap and Paul A Schilpp. *The Philosophy of Rudolf Carnap*. Cambridge University Press Cambridge, 1963.

[27] Davide Castelvecchi. Can we open the black box of ai? *Nature News*, 538(7623):20, 2016.

[28] Taolue Chen, Giuseppe Primiero, Franco Raimondi, and Neha Rungta. A computationally grounded, weighted doxastic logic. *Studia Logica*, 104(4):679–703, 2016.

[29] Giorgio Cignarale and Giuseppe Primiero. A multi-agent depth bounded boolean logic. In *Software Engineering and Formal Methods. SEFM 2020 Collocated Workshops: ASYDE, CIFMA, and CoSim-CPS, Amsterdam, The Netherlands, September 14–15, 2020, Revised Selected Papers*, pages 176–191. Springer, 2021.

[30] Edmund M Clarke and E Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on logic of programs*, pages 52–71. Springer, 1981.

[31] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[32] Joseph L Cowan. Wittgenstein's philosophy of logic. *The philosophical review*, 70(3):362–375, 1961.

[33] Carl F Craver and Lindley Darden. *In search of mechanisms: Discoveries across the life sciences*. University of Chicago Press, 2013.

[34] Kathleen A Creel. Transparency in complex computational systems. *Philosophy of Science*, 87(4):568–589, 2020.

[35] Max J Cresswell. Intensional logics and logical truth. *Journal of Philosophical Logic*, pages 2–15, 1972.

[36] Adnan Darwiche and Auguste Hirth. On the reasons behind decisions. *arXiv preprint arXiv:2002.09284*, 2020.

[37] Fabio A D'Asaro, Matteo Spezialetti, Luca Raggioli, and Silvia Rossi. Towards an inductive logic programming approach for explaining black-box preference learning systems. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, volume 17, pages 855–859, 2020.

[38] Fabio Aurelio D'Asaro, Paolo Baldi, Giuseppe Primiero, et al. Introducing k-lingo: a k-depth bounded version of asp system clingo. In *KR2021*, pages 661–665. IJCAI Organization, 2021.

[39] C. Daws. Symbolic and parametric model checking of discrete-time markov chains. In L. Zhiming and K. Araki, editors, *Theoretical Aspects of Computing - ICTAC 2004, First International Colloquium, Guiyang, China, September 20-24, 2004, Revised Selected Papers*, volume 3407 of *Lecture Notes in Computer Science*, pages 280–294, Cham, 2004. Springer.

[40] G. De Cooman, J. De Bock, and S. Lopatatzidis. Imprecise stochastic processes in discrete time: global models, imprecise markov chains and ergodic theorems. *Int. J. Approx. Reason.*, 76:18–46, 2016.

[41] Gert De Cooman and Filip Hermans. Imprecise probability trees: Bridging two theories of imprecise probability. *Artificial Intelligence*, 172(11):1400–1427, 2008.

[42] Gert De Cooman, Filip Hermans, and Erik Quaeghebeur. Imprecise markov chains and their limit behavior. *Probability in the Engineering and Informational Sciences*, 23(4):597–635, 2009.

[43] Bruno De Finetti. La prévision: ses lois logiques, ses sources subjectives. In *Annales de l'institut Henri Poincaré*, volume 7, pages 1–68, 1937.

[44] Pierre Simon de Laplace. *Théorie analytique des probabilités*, volume 7. Courcier, 1820.

[45] Richard A De Millo, Richard J Lipton, and Alan J Perlis. Social processes and proofs of theorems and programs. *Communications of the ACM*, 22(5):271–280, 1979.

[46] Augustus De Morgan. *Formal logic: or, the calculus of inference, necessary and probable*. Taylor and Walton, 1847.

[47] Henk W de Regt. *Understanding Scientific Understanding*. Oxford University Press, Oxford, UK, 2017.

[48] Karina V Delgado, Leliane N De Barros, Daniel B Dias, and Scott Sanner. Real-time dynamic programming for Markov decision processes with imprecise probabilities. *Artificial Intelligence*, 230:192–223, 2016.

[49] Lorenz Demey, Barteld Kooi, and Joshua Sack. Logic and Probability. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2019 edition, 2019.

[50] Sébastien Destercke and Gert de Cooman. Relating epistemic irrelevance to event trees. In *Soft Methods for Handling Variability and Imprecision*, pages 66–73. Springer, Cham, 2008.

[51] Edsger Wybe Dijkstra, Edsger Wybe Dijkstra, Edsger Wybe Dijkstra, and Edsger Wybe Dijkstra. *A discipline of programming*, volume 613924118. prentice-hall Englewood Cliffs, 1976.

[52] Rolf Drechsler et al. *Advanced formal verification*, volume 122. Springer, 2004.

[53] Ho Ngoc Duc. Logical omniscience vs. logical ignorance on a dilemma of epistemic logic. In Carlos A. Pinto-Ferreira and Nuno J. Mamede, editors, *Progress in Artificial Intelligence, 7th Portuguese Conference on Artificial Intelligence, EPIA '95, Funchal, Madeira Island, Portugal, October 3-6, 1995, Proceedings*, volume 990 of *Lecture Notes in Computer Science*, pages 237–248. Springer, 1995.

[54] Ho Ngoc Duc. Reasoning about rational, but not logically omniscient, agents. *Journal of Logic and Computation*, 7(5):633–648, 1997.

[55] Ho Ngoc Duc. *Resource bounded reasoning about knowledge*. PhD thesis, Leipzig University, Germany, 2001.

[56] Juan M Durán and Nico Formanek. Grounds for trust: Essential epistemic opacity and computational reliabilism. *Minds and Machines*, 28(4):645–666, 2018.

[57] Juan Manuel Durán and Karin Rolanda Jongsma. Who is afraid of black box algorithms? on the epistemological and ethical basis of trust in medical ai. *Journal of Medical Ethics*, 47(5):329–335, 2021.

[58] Marcello D'Agostino. Depth-bounded logic for realistic agents. *L&PS–Logic & Philosophy of Science*, 11(1):3–57, 2013.

[59] Marcello D'Agostino, Marcelo Finger, and Dov Gabbay. Semantics and proof-theory of depth bounded boolean logics. *Theoretical Computer Science*, 480:43–68, 2013.

[60] Jennifer Elgot-Drapkin, Sarit Kraus, Michael Miller, Madhura Nirkhe, and Donald Perlis. Active logics: A unified formal approach to episodic reasoning. Technical report, 1999.

[61] Alessandro Facchini and Alberto Termine. Intelligenza artificiale e metodo scientifico: può l'ia sostituire l'intuizione umana nel processo di scoperta scientifica? *Rivista teologica di Lugano*, 26(3):11–28, 2021.

[62] Alessandro Facchini and Alberto Termine. Towards a taxonomy for the opacity of ai systems. In *Philosophy and Theory of Artificial Intelligence 2021*, pages 73–89. Springer, 2022.

[63] Alessandro Facchini and Alberto Termine. Beyond hypothesis-driven and data-driven biology through explainable ai: a proposal, unpublished.

[64] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*. MIT press, 2004.

[65] Andrea Ferrario, Michele Loi, and Eleonora Viganò. In ai we trust incrementally: A multi-layer model of trust to analyze human-artificial intelligence interactions. *Philosophy & Technology*, 33:523–539, 2020.

[66] James H Fetzer. Program verification: The very idea. *Communications of the ACM*, 31(9):1048–1063, 1988.

[67] Bruno de Finetti. Foresight: Its logical laws, its subjective sources. In *Breakthroughs in statistics*, pages 134–174. Springer, 1992.

[68] Ionut Florescu and Ciprian A Tudor. *Handbook of probability*. John Wiley & Sons, 2013.

[69] Robert W Floyd. Assigning meanings to programs. *Program Verification: Fundamental Issues in Computer Science*, pages 65–81, 1993.

[70] Alex A Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10, 2014.

[71] Nir Fresco and Giuseppe Primiero. Miscomputation. *Philosophy & Technology*, 26:253–272, 2013.

[72] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[73] Marcello Frixione. *Logica, significato e intelligenza artificiale*. F. Angeli, 1994.

[74] Warren Goldfarb. Frege's conception of logic. *Future pasts: The analytic tradition in twentieth-century philosophy*, pages 25–41, 2001.

[75] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2019.

[76] Riccardo Guidotti, Anna Monreale, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[77] Marta Halina. Mechanistic explanation and its limits. In Stuart Glennan and Phyllis Illari, editors, *The Routledge handbook of mechanisms and mechanical philosophy*, pages 213–224. Routledge, 2017.

[78] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[79] Mihály Héder. The epistemic opacity of autonomous systems and the ethical consequences. *AI & SOCIETY*, pages 1–9, 2020.

[80] Filip Hermans and Damjan Škulj. Stochastic processes. *Introduction to Imprecise Probabilities*, pages 258–278, 2014.

[81] Jaakko Hintikka. Impossible possible worlds vindicated. In *Game-theoretical semantics*, pages 367–379. Springer, 1979.

[82] Kaarlo Jaakko Juhani Hintikka. Knowledge and belief: An introduction to the logic of the two notions. 1962.

[83] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[84] Charles Antony Richard Hoare. The mathematics of programming. In *Foundations of Software Technology and Theoretical Computer Science: Fifth Conference, New Delhi, India December 16–18, 1985 Proceedings 5*, pages 1–18. Springer, 1985.

[85] Colin Howson and Peter Urbach. *Scientific reasoning: the Bayesian approach*. Open Court Publishing, 2006.

[86] Paul Humphreys. The philosophical novelty of computer simulation methods. *Synthese*, 169(3):615–626, 2009.

[87] Paul Humphreys. The philosophical novelty of computer simulation methods. *Synthese*, 169(3):615–626, 2009.

[88] Alan Hájek. Interpretations of Probability. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2019 edition, 2019.

[89] PK Illari and Jon Williamson. Mechanisms are real and local. In Phyllis McKay Illari, Federica Russo, and Jon Williamson, editors, *Causality in the Sciences*, pages 818–844. Oxford University Press, 2011.

[90] Samin S Ishtiaq and Peter W O'Hearn. Bi as an assertion language for mutable data structures. In *Proceedings of the 28th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 14–26, 2001.

[91] Ulf Johansson, Rikard König, and Lars Niklasson. The truth is in there-rule extraction from opaque models using genetic programming. In *FLAIRS Conference*, pages 658–663. Miami Beach, FL, 2004.

[92] John D Kelleher. *Deep learning*. MIT press, 2019.

[93] Rob Kitchin and Gavin McArdle. What makes big data, big data? exploring the ontological characteristics of 26 datasets. *Big Data & Society*, 3(1):2053951716631130, 2016.

[94] A.N. Kolomogoroff. *Grundbegriffe der wahrscheinlichkeitsrechnung*. Ergebnisse Der Mathematik, 1933.

[95] Kurt Konolige. *A deduction model of belief and its logics*. PhD thesis, Stanford University, USA, 1984.

[96] Thomas Krak, Jasper De Bock, and Arno Siebes. Imprecise continuous-time Markov chains. *International Journal of Approximate Reasoning*, 88:452–528, 2017.

[97] Thomas E. Krak, Natan T'Joens, and Jasper De Bock. Hitting times and probabilities for imprecise markov chains. In Jasper De Bock, Cassio P. de Campos, Gert de Cooman, Erik Quaeghebeur, and Gregory R. Wheeler, editors, *International Symposium on Imprecise Probabilities: Theories and Applications, ISIPTA 2019, 3-6 July 2019, Thagaste, Ghent, Belgium*, volume 103 of *Proceedings of Machine Learning Research*, pages 265–275, Ghent, 2019. PMLR.

[98] Saul Kripke. A completeness theorem in modal logic. *J. Symb. Log.*, 24(1):1–14, 1959.

[99] Maya Krishnan. Against interpretability: a critical examination of the interpretability problem in machine learning. *Philosophy & Technology*, 33(3):487–502, 2020.

[100] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[101] Adam La Caze. Frequentism. In *The Oxford handbook of probability and philosophy*. 2016.

[102] Imre Lakatos. *Proofs and refutations*. Nelson London, 1963.

[103] Costanza Larese. *The Principle of Analyticity of Logic, a Philosophical and Formal Perspective*. PhD thesis, Scuola Normale Superiore di Pisa, 2019.

[104] Marco Larotonda and Giuseppe Primiero. A depth-bounded semantics for becoming informed. In *Software Engineering and Formal Methods. SEFM 2022 Collocated Workshops: AI4EA, F-IDE, CoSim-CPS, CIFMA, Berlin, Germany, September 26–30, 2022, Revised Selected Papers*, pages 366–382. Springer, 2023.

[105] Gregory F Lawler. *Introduction to stochastic processes*. Chapman and Hall/CRC, London, UK, 2018.

[106] Ronald J Leach. *Introduction to software engineering*. Chapman and Hall/CRC, 2018.

[107] Hannes Leitgeb and André Carus. Rudolf Carnap. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Summer 2023 edition, 2023.

[108] Sabina Leonelli. *Data-centric biology: A philosophical study*. University of Chicago Press, 2016.

[109] Hector J Levesque. A logic of implicit and explicit belief. In *AAAI*, pages 198–202, 1984.

[110] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable AI: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2021.

[111] Charles H Lindsey. A history of algol 68. In *History of programming languages—II*, pages 27–96. 1996.

[112] A. Lomuscio and F. Raimondi. MCMAS: A model checker for multi-agent systems. In Holger Hermanns and Jens Palsberg, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 12th International Conference, TACAS 2006 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2006, Vienna, Austria, March 25 - April 2, 2006, Proceedings*, volume 3920 of *Lecture Notes in Computer Science*, pages 450–454, Cham, 2006. Springer.

[113] Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. Mcmas: A model checker for the verification of multi-agent systems. In *International conference on computer aided verification*, pages 682–688. Springer, 2009.

[114] Ezequiel López-Rubio and Emanuele Ratti. Data science and molecular biology: prediction and mechanistic explanation. *Synthese*, 198(4):3131–3156, 2021.

[115] Bernard Marr. *Big Data: Using SMART big data, analytics and metrics to make better decisions and improve performance*. John Wiley & Sons, London, 2015.

[116] Denis Deratani Mauá, Cassio Polpo de Campos, Alessio Benavoli, and Alessandro Antonucci. Probabilistic inference in credal networks: New complexity results. *Journal of Artificial Intelligence Research*, 50:603–637, 2014.

[117] Sally I. McClean, B. McAlea, and Peter H. Millard. Using a Markov reward model to estimate spend-down costs for a geriatric department. *J. Oper. Res. Soc.*, 49(10):1021–1025, 1998.

[118] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[119] Enrique Miranda and Gert De Cooman. Structural judgements. *Introduction to imprecise probabilities*, pages 56–78, 2014.

[120] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg

Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[121] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[122] Peter W O'Hearn and David J Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, 5(2):215–244, 1999.

[123] Andrés Páez. The pragmatic turn in explainable artificial intelligence (xai). *Minds and Machines*, 29(3):441–459, 2019.

[124] Cecilia Panigutti, Alan Perotti, and Dino Pedreschi. Doctor xai: an ontology-based approach to black-box sequential data classification explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 629–639, 2020.

[125] Judea Pearl. *Causality*. Cambridge University Press, 2 edition, 2009.

[126] Judea Pearl. The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60, 2019.

[127] Judea Pearl, Madelyn Glymour, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, London, 2016.

[128] Judea Pearl and Dana Mackenzie. *The Book of Why: The New Science of Cause and Effect*. Hachette, London, 2018.

[129] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. ieee, 1977.

[130] Karl R Popper. The propensity interpretation of probability. *The British journal for the philosophy of science*, 10(37):25–42, 1959.

[131] Giuseppe Primiero. *On the foundations of computing*. Oxford University Press, 2019.

[132] Stathis Psillos. *Scientific realism: How science tracks truth*. Routledge, London, 2005.

[133] W. V. Quine. The roots of reference. *British Journal for the Philosophy of Science*, 27(1):93–96, 1976.

[134] Frank P Ramsey. Truth and probability. In *Readings in formal epistemology*, pages 21–45. Springer, 2016.

[135] B Randell and JN Buxton. Software engineering techniques: Report of a conference sponsored by the nato science committee, rome, italy, 27th-31st october 1969. 1970.

[136] Mattias Skipper Rasmussen. Dynamic epistemic logic and logical omniscience. *Logic and Logical Philosophy*, 24(3), 2015.

[137] Hans Reichenbach. *The theory of probability*. Univ of California Press, 1971.

[138] Hans Reichenbach. *The direction of time*, volume 65. Univ of California Press, 1991.

[139] John C Reynolds. Separation logic: A logic for shared mutable data structures. In *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*, pages 55–74. IEEE, 2002.

[140] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[141] Felipe Romero. Philosophy of science and the replicability crisis. *Philosophy Compass*, 14(11):e12633, 2019.

[142] Sheldon M Ross. *Introduction to probability and statistics for engineers and scientists*. Elsevier, 2004.

[143] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *CoRR*, abs/2103.11251, 2021.

[144] Bertrand Russell. *Introduction to mathematical philosophy*. Taylor & Francis, 2022.

[145] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature, Cham, 2019.

[146] Dana Scott. *Outline of a mathematical theory of computation*. Oxford University Computing Laboratory, Programming Research Group Oxford, 1970.

[147] Dana S Scott and Christopher Strachey. *Toward a mathematical semantics for computer languages*, volume 1. Oxford University Computing Laboratory, Programming Research Group Oxford, 1971.

[148] Glenn Shafer and Vladimir Vovk. *Probability and finance: it's only a game!*, volume 491. John Wiley & Sons, London, UK, 2005.

[149] Jan Sprenger and Stephan Hartmann. *Bayesian philosophy of science*. Oxford University Press, 2019.

[150] Robert Stalnaker. The problem of logical omniscience, i. *Synthese*, pages 425–440, 1991.

[151] Richard Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, 9(1):67–72, 1979.

[152] Mauricio Suárez. *Philosophy of probability and statistical modelling*. Cambridge University Press, 2020.

[153] Emily Sullivan. Understanding from machine learning models. *The British Journal for the Philosophy of Science*, 2020.

[154] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, Cambridge, MA, 1998.

[155] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

[156] Matti Tedre. *The science of computing: shaping a discipline*. CRC Press, 2014.

[157] A. Termine, A. Antonucci, A. Facchini, and G. Primiero. Robust model checking with imprecise markov reward models. In *International Symposioum on Imprecise Probabilities: Theories and Applications, ISIPTA 2021, July 6-9 2021, Granada, Spain*, Proceedings of Machine Learning Research, Granada, 2021. PMLR.

[158] Alberto Termine, Alessandro Antonucci, Giuseppe Primiero, and Alessandro Facchini. Logic and model checking by imprecise probabilistic interpreted systems. In *European Conference on Multi-Agent Systems*, pages 211–227. Springer, 2021.

[159] Alberto Termine, Alessandro Antonucci, Giuseppe Primiero, and Alessandro Facchini. Imprecise probabilistic model checking for stochastic multi-agent systems. Forthcoming in SN Computer Science, Forthcoming.

[160] Alberto Termine, Giuseppe Primiero, and Fabio Aurelio D'Asaro. Modelling accuracy and trustworthiness of explaining agents. In *Logic, Rationality, and Interaction: 8th International Workshop, LORI 2021, Xi'ian, China, October 16-18, 2021, Proceedings 8*, pages 232–245. Springer, 2021.

[161] N. T'Joens, T.E. Krak, J. De Bock, and G. De Cooman. A recursive algorithm for computing inferences in imprecise markov chains. In G. Kern-Isberner and Z. Ognjanovic, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 15th European Conference, ECSQARU 2019, Belgrade, Serbia, September 18-20, 2019, Proceedings*, volume 11726 of *Lecture Notes in Computer Science*, pages 455–465, Cham, 2019. Springer.

[162] Isaac Todhunter. *History of the Mathematical Theory of Probability from the time of Pascal to that of Laplace*. Macmillan and Company, 1865.

[163] Matthias C. M. Troffaes and Damjan Škulj. Model checking for imprecise Markov chains. In Fabio Cozman, Thierry Denoeux, Sebastien Destercke, and Teddy Seidenfeld, editors, *ISIPTA '13 : Proceedings of the Eighth International Symposium on Imprecise Probability: Theories and Applications*, pages 337–344. Society for Imprecise Probability: Theories and Applications (SIPTA), July 2013.

[164] Raymond Turner. Computational artifacts. In *Computational Artifacts*, pages 25–29. Springer, Cham, 2018.

[165] Natan T'Joens and Jasper De Bock. Global upper expectations for discrete-time stochastic processes: in practice, they are all the same! In *International Symposium on Imprecise Probability: Theories and Applications*, pages 310–319. PMLR, 2021.

[166] Alasdair Urquhart. The undecidability of entailment and relevant implication. *The Journal of Symbolic Logic*, 49(4):1059–1073, 1984.

[167] Hans van Ditmarsch, Wiebe van der Hoek, Joseph Y. Halpern, and Barteld Kooi, editors. *Handbook of Epistemic Logic*. College Publications, London, 2015.

[168] Hans Van Ditmarsch, Wiebe van Der Hoek, and Barteld Kooi. *Dynamic epistemic logic*, volume 337. Springer Science & Business Media, 2007.

[169] Alexandros Vassiliades, Nick Bassiliades, and Theodore Patkos. Argumentation and explainable artificial intelligence: a survey. *The Knowledge Engineering Review*, 36, 2021.

[170] John Venn. The logic of chance, an essay on the theory of probability. 1876.

[171] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

[172] Richard Von Mises. *Probability, statistics, and truth*. Courier Corporation, 1981.

[173] Peter Walley. *Statistical reasoning with imprecise probabilities*. Chapman and Hall, New York, 1991.

[174] Peter Walley. Towards a unified theory of imprecise probability. *Int. J. Approx. Reason.*, 24(2-3):125–148, 2000.

[175] Wei Wan, Jamal Bentahar, and Abdessamad Ben Hamza. Model checking epistemic-probabilistic logic using probabilistic interpreted systems. *Knowledge Based Systems*, 50:279–295, 2013.

[176] Eric Wang, Pasha Khosravi, and Guy Van den Broeck. Probabilistic sufficient explanations. *arXiv preprint arXiv:2105.10118*, 2021.

[177] Heinrich Wansing. A general possible worlds framework for reasoning about knowledge and belief. *Studia logica*, 49(4):523–539, 1990.

[178] Richard Zach. Hilbert's Program. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2019 edition, 2019.

[179] Carlos Zednik. Solving the black box problem: a normative framework for explainable artificial intelligence. *Philosophy & Technology*, pages 1–24, 2019.