1
Introduction

Temporal logics provide a generic logical framework for modelling and reasoning about time and temporal aspects of the world. While stemming from philosophical considerations and discussions, temporal logics have become over the past 50 years very useful and important in computer science, and particularly for formal specification, verification and synthesis of computerised systems of various nature: sequential, concurrent, reactive, discrete, real time, stochastic, etc. This book provides a comprehensive exposition of the most popular discrete-time temporal logics, used for reasoning about transition systems and computations in them.

1.1 Temporal Logics and Computer Science: A Brief Overview

We begin with a brief historical overview of temporal reasoning and logics and their role in computer science.

1.1.1 Historical Origins of Temporal Reasoning and Logics

The study of time and temporal reasoning goes back to the antiquity. For instance, some of Zeno's famous paradoxical arguments refer to the nature of time and the question of infinite divisibility of time intervals. Perhaps the earliest scientific reference on temporal reasoning, however, is Aristotle's '*Sea Battle Tomorrow*' argument in the *Organon II – On*

Interpretation, Chapter 9, claiming that *future contingents*, i.e. statements about possible future events which may or may not occur, such as 'There will be a sea-battle tomorrow', should not be ascribed definite truth values at the present time. A few decades later the philosopher Diodorus Cronus from the Megarian school illustrated the problem of future contingents in his famous *Master Argument* where he, inter alia, defined 'possible' as '*what is or will ever be*' and 'necessary' as '*what is and will always be*'.

Philosophical discussions on temporality, truth, free will and determinism, and their relationships continued during the Middle ages. In particular, William of Ockham held that propositions about the contingent future cannot be known by humans as true or false because only God knows their truth value now. However, he argued that humans can still freely choose amongst different possible futures, thus suggesting the idea of a future-branching model of time with many possible time-lines (histories), truth of propositions being relativised to a possible actual history. This model of time is now often called *Ockhamist* or *actualist*. Later, several philosophers and logicians raised and analysed problems relating temporality with nondeterminism, historical necessity, free will, God's will and knowledge, etc., proposing various different solutions. Still, little tangible progress in the formal study of temporal reasoning occurred until some new ideas emerged in the late 19th to early 20th century, including the developments of Minkowski's four-dimensional space time model and its application to Einstein's relativity theory, of Reichenbach's theory of temporality and tenses in natural language, as well as some informal philosophical studies of temporal reasoning by C. Peirce, McTaggart, B. Russell, W. Quine, J. Findley, J. Łukasiewicz and others. However, temporal logic as a formal logical system only emerged in the early 1950s when the philosopher Arthur Prior set out to analyse and formalise such arguments, leading him, inter alia, to the invention of (as Prior called it) Tense Logic, of which he developed and discussed several formal systems. Prior's seminal work initiated the modern era of temporal logical reasoning, leading to numerous important applications not only to philosophy, but also to computer science, artificial intelligence and linguistics.

Prior introduced the following basic temporal modalities:

P 'It *has at some time in the past been* the case that …'
F 'It *will at some time in the future be* the case that …'
H 'It *has always been* the case that …'
G 'It *will always be* the case that …'

Put together in a formal logical language they allow complex temporal patterns to be expressed, for instance:

$$\textit{There is a solstice} \wedge \textit{there is a lunar eclipse} \rightarrow \mathsf{GP}\ \text{'}\textit{There is a solstice and a lunar eclipse}\text{'}$$

means that if there is a solstice and there is a lunar eclipse now then it will always be the case that there has been a solstice and a lunar eclipse at the same moment of time.

Subsequently, other temporal modalities were introduced, notably *Nexttime* and the binary operators *Since* and *Until* in Kamp's (1968) very influential doctoral thesis. The

simplest and most common models of time are linear-ordered time flows and Prior's temporal operators, as well as Since and Until, have natural interpretation in them. The resulting temporal logics for linear time are quite expressive. A classical expressiveness result, due to Kamp (1968) states that the temporal logic with Since and Until is as expressive as first-order logic. The temporal logic with operators for Next-time and Until interpreted on the time flow of natural numbers, known as the *linear-time temporal logic* LTL, became since the late 1970s the most popular temporal logic used in computer science.

Coming back to Prior's philosophical studies of temporal reasoning: in his analysis of Diodorus's Master Argument Prior argued that its fallacy lay in Diodorus's assumption that whatever is, or is not, or will be, or will never be, has in the past been necessarily so – thus, in effect assuming that the future is deterministic. Prior supported Aristotle's view that 'while it is now beyond the power of men or gods to affect the past, there are alternative futures between which choice is possible'. In order to resolve the problems pointed out by Aristotle and Diodorus, Prior wanted, inter alia, to capture the *logic of historical necessity*. His philosophical analysis and the quest for formalisation of the arguments for 'the incompatibility of foreknowledge (and fore-truth) and indeterminism' lead him to consider two formalisations of temporal logic of branching time, reflecting the 'Peircean' and the 'Ockhamist' (or, 'actualist') views, underlying respectively the ideas behind the *branching-time temporal logics* CTL and CTL* presented here. For further reading on the history of temporal reasoning and logics, see Øhrstrøm and Hasle (1995). In particular, for details on Prior's views and motivating analyses, see Prior (1967, Chapter VII) and Øhrstrøm and Hasle (1995, Chapters 2.6 and 3.2). A broad but concise overview of temporal logics is Goranko and Galton (2015).

1.1.2 The Role of Temporal Logics in Computer Science

Temporal aspects and phenomena are pervasive in computer and information systems. These include: scheduling of the execution of programs by an operating system; concurrent and reactive systems and, in particular, synchronisation of concurrent processes; real-time processes and systems; hardware verification; temporal databases, etc. Many of these are related to specification and verification of properties of *transition systems* and *computations* in them. Formally, transition systems are directed graphs consisting of states and transitions between them. They are used to model sequential and concurrent processes. There can be different types of transitions (e.g. affected by different types of actions) which we indicate by assigning different labels to them. Thus, more generally, we talk about *labelled transition systems*. A state in a transition system may satisfy various properties: it can be *initial, terminal, deadlock, safe or unsafe*, etc. One can describe state properties by formulae of a suitable state-description language; on propositional level these are simply atomic propositions. The set of such propositions that are true at a given state is encoded in the *label* of that state, and a transition system where every state is assigned such label will be called an *interpreted transition system*. In terms of the semantics of modal logics, transition

systems are simply Kripke frames, and the labelling of states corresponds to a valuation of the atomic propositions in such frames, so interpreted transition systems are just Kripke models. A *computation* in a transition system is, intuitively, the observable trace of a run – a sequence of states produced by following the transition relations in the system, viz. the sequence of the labels of these states. It can be regarded as a record of all observable successive intermediate results of the computing process.

Early work implicitly suggesting applications of temporal reasoning to modelling and analysis of deterministic and stochastic transition systems is the theory of processes and events in Rescher and Urquhart (1971, Chapter XIV). The use of temporal logic for specification and verification of important properties of reactive and concurrent systems, such as *safety*, *liveness* and *fairness*, was first explicitly proposed in Amir Pnueli's (1977) seminal paper (see also Abrahamson 1979; Lamport 1980; Ben-Ari et al. 1981, 1983; Clarke and Emerson 1981). In particular, Pnueli proposed and developed, together with Zohar Manna, versions of the temporal linear-time logic LTL, in Manna and Pnueli (1979, 1981) as logical framework for deductive verification of such systems. Other influential early works on temporal logics of programs and processes include Abrahamson (1979, 1980) and Kröger (1987).

Since the late 1970s temporal logics have found numerous other applications in computer science. The key for their success and popularity is that temporal logics are syntactically simple and elegant, have natural semantics in interpreted transition systems, are well expressive for properties of computations and – very importantly – have good computational behaviour. Thus, they provide a very appropriate logical framework for formal specification and verification of programs and properties of transition systems. Depending on the type of systems and properties to specify and verify, two major families of temporal logics have emerged: *linear-time and branching-time logics*. Manna and Pnueli (1992) is a comprehensive reference on the early use of (linear time) temporal logics for specification of concurrent and reactive systems, and Manna and Pnueli (1995) is its continuation showing how that can be used to guarantee safety of such systems.

Two major developments, both starting in the 1980s, contributed strongly to the popularity and success of temporal logics in computer science. The first one is the advancement of *model checking* as a method for formal verification by Clarke and Emerson (1981), followed by Clarke et al. (1983, 1986), and independently by Queille and Sifakis (1982a). Introductions to model checking can be found in Huth and Ryan (2000) and Clarke and Schlingloff (2001), and for more comprehensive expositions, see Clarke et al. (2000) and Baier and Katoen (2008). The second major development is the emergence of *automata-based methods* for verification, initially advocated in a series of papers by Streett (1982), Sistla, Vardi and Wolper (1987), Vardi and Wolper (1986a,b) and Emerson and Sistla (1984), and further developed in Sistla et al. (1987), Emerson and Jutla (1988), Muller et al. (1988), Streett and Emerson (1989), Thomas (1990), Emerson and Jutla (1991), Vardi (1991), Vardi and Wolper (1994), Vardi (1996, 1997), Wolper (2000), Kupferman et al. (2000), Löding and Thomas (2000), etc. See also Vardi and Wilke (2007), Vardi (2007) and Grädel et al. (2002) for broader overviews and further references.

1.1.3 The Influence of Computer Science on the Development of Temporal Logics

The applications of temporal logics for specification and verification of computer systems have in turn strongly stimulated the study of their expressiveness and computational complexities and the development of efficient algorithmic methods for solving their basic logical decision problems.

The first proof systems developed for temporal logics were Hilbert-style axiomatic systems (see e.g. Rescher and Urquhart 1971; Goldblatt 1992; Reynolds 2001), as well as tableaux-style calculi and some Gentzen-type systems (see e.g. Fitting 1983; Wolper 1985; Goré 1991; Schwendimann 1998b; Goré 1999). In particular, efficient and intuitively appealing tableau-based methods for satisfiability testing of temporal formulae were developed in the early 1980s, by Wolper (1983, 1985) for the linear-time logic LTL, Ben-Ari, Manna and Pnueli (1981) for the branching-time logic UB and Emerson and Halpern (1982, 1985) for the branching-time logic CTL. The increasing demand coming from the field of formal verification for efficient algorithmic methods solving verification problems has led to the subsequent development of new methods, based on automata (see earlier references) and games (Stirling 1995, 1996; Stevens and Stirling 1998; Lange and Stirling 2000, 2001, 2002; Grädel 2002; Lange 2002a; see also Grädel et al. 2002 for a comprehensive coverage and further references). Other methods, such as temporal resolution, have also been developed recently (see e.g. Fisher 2011).

Whereas traditional systems of logical deduction and tableaux-style calculi can be qualified as direct methods, since they handle temporal formulae directly, the automata-based and the game-theoretic approaches work by reducing the decision problems for temporal logics to decision problems about automata and games, respectively. For instance, the automata-based approach is based on reducing logical to automata-based decision problems in order to take advantage of known results from automata theory (D'Souza and Shankar 2012). The most standard target problems on automata used in this approach are the *language nonemptiness problem* (checking whether an automaton admits at least one accepting computation) and the *language inclusion problem* (checking whether the language accepted by an automaton $\mathcal{A}$ is included in the language accepted by another automaton $\mathcal{B}$). In a pioneering work, Richard Büchi (1962) introduced a class of automata on infinite words and showed that these automata are equivalent in a precise sense to formulae in the monadic second-order theory (MSO) of the structure $(\mathbb{N}, <)$, which eventually resulted in Büchi's proof of the decidability of that theory. Later this idea and the result were extended by Rabin (1969) in his seminal paper to automata on trees and the decidability of the MSO theory of the infinite binary tree (equivalently, the MSO theory of two successor functions). These results provide the theoretical foundation of the automata-based decision methods in temporal logics.

Likewise, in the game-theoretic approach, the question of whether a temporal formula is satisfiable corresponds to the question of whether some designated player has a winning strategy in an associated satisfiability game. More generally, the existence of winning strategies of a given player in model-checking games and satisfiability-checking games provide

uniform characterisations of the model checking and the satisfiability-testing problems for many temporal logics. These game-theoretic characterisations are of particular interest in the context of program verification.

As we demonstrate in this book, the methods for solving decision problems of temporal logics, based on automata, tableaux and games are closely related, both conceptually and technically, while each of them has pros and cons compared to the other. Because of the conceptual elegance and technical power and convenience, the automata-based methods have generally been favoured by the researchers in the area of verification and most of the tools that have been implemented are based on automata. On the other hand, tableau-based methods for model checking and satisfiability checking of temporal logics have so far been less developed for practical purposes and less tested for industrial applications, but are arguably more natural and intuitive from logical perspective, easier for execution by humans and potentially more flexible and practically efficient, if suitably optimised.

1.2 Structure and Summary of the Book Content

With this book we aim to offer a comprehensive, uniform, technically precise and conceptually in-depth exposition of the field, focusing on the intrinsically logical aspects of it and including both classical and recent results and methods of fundamental importance. The book has been written with the specific intention to be suitable both as a comprehensive graduate textbook and as a professional reference on the current state of the art in the field. For that purpose, it presents an essentially self-contained and rigorous treatment of the content, with precise definitions, statements and many detailed proofs, as well as numerous examples and exercises. Many easy or routine proofs are omitted in the main text but are explicitly left as exercises.

Except for the introductory chapter, we do not make references and credits to specific results in the main text, but provide such references in the bibliographic notes at the end of every chapter, where we also mention related topics not treated in the book and provide a number of additional references.

The book consists of 15 chapters structured in four parts. Here we will give a concise summary of their contents and will mention a few highlights in each chapter. More detailed summaries can be found at the beginning of each chapter.

Part I, 'Models', presents the basic theory of the abstract models for the temporal logics studied further, viz. transition systems and computations in them.

Chapter 2, 'Preliminaries and Background I', provides some preliminary material that will be needed further, including basics of sets, binary relations and orders, fixpoint theory, computational complexity and 2-player games on graphs. We do not intend to teach this material here but rather to recall the most basic definitions, terminology and notation, mainly for readers' convenience, as a quick reference.

Chapter 3, 'Transition Systems', introduces the basic concepts and facts related to transition systems, computations and important types of their properties. In particular, we provide simple algorithms solving the basic reachability problems in transition systems. Besides, we discuss transition systems as abstract models of the behaviour of real systems with respect to transitions between states. A real transition system can be modelled by different abstract transition systems and at different levels of detail. The chapter addresses the important question of when two transition systems should be considered behaviourally equivalent, that is, modelling essentially equivalent real transition systems. That question leads to the fundamental notion of bisimulation, the main versions of which are introduced and studied in the chapter. In particular, we introduce bisimulation games and relate the existence of winning strategies for one of the players in such games to the existence of bisimulation between the transitions systems on which the game is played. This is the first place in the book where games are considered and they will be one of its main themes.

Part II, 'Logics', is the core part of the book, presenting and studying the most important temporal logics used for specification and verification of discrete transition systems and many variations of them. In a relatively uniform manner we introduce the syntax and semantics of each of these logical systems, discuss and illustrate their use for formal specification of properties of transition systems and computations, study their expressiveness and the basic logical decision problems (model checking, satisfiability and validity testing) for them, relate them to standard verification tasks and present algorithms for solving some of these decision problems. The expositions are written from the primary perspective of computer science rather than from the perspective of pure modal and temporal logics. Thus, we have emphasised the more relevant topics associated with transition systems, such as expressiveness, bisimulation invariance, model checking, small model property and deciding satisfiability, while other fundamental logical topics, such as deductive systems and proof theory (except for short sections on axiomatic systems and derivations in them), model theory, correspondence theory, algebraic semantics, duality theory, etc. are almost left untouched here, but references are given in the bibliographic notes to other books where they are treated in depth.

Following is a brief summary of the chapters in this part.

Chapter 4, 'Preliminaries and Background II', provides some common background, terminology and notation related to logical decision problems and deductive systems. It assumes some knowledge of propositional logic, upon which all temporal logics studied here are built.

Chapter 5, 'Basic Modal Logics', is a concise introduction to the multimodal logic BML, regarded here as the basic temporal logic for reasoning about interpreted transition systems. Highlights on that chapter include: invariance of BML formulae under bisimulations and characterising the existence of bounded bisimulations between interpreted transition systems with BML formulae; study of the logical decision problems of model checking and

satisfiability testing for BML, model-checking games and an axiomatic system for BML. Inter alia, we provide a simple optimal algorithm for testing satisfiability in BML that runs in polynomial space. These are further adapted to the extension BTL of BML with past-time operators. This chapter can also be viewed as a stepping stone towards the more expressive and interesting branching-time temporal logics CTL, CTL* and the modal μ-calculus, presented further.

Chapter 6, 'Linear-Time Temporal Logics', is devoted to temporal logics for linear-time models, representing single computations (i.e. infinite sequences of states generated by the transition relations), rather than entire transition systems. Here we present and study in detail the linear-time temporal logic LTL. We focus again on the logical and computational properties of this logic, viz. its semantics, expressiveness, model checking and testing of satisfiability and validity. We provide conceptually simple decision procedures for satisfiability testing and model-checking problems for LTL based on the so-called ultimately periodic model property and discuss how these can be refined and transformed into optimal decision procedures. (Alternative decision methods, essentially using the same property but based on tableaux and automata, are developed further, respectively in Chapters 13 and 14.) We also present and discuss here some of the most interesting extensions of LTL over linear models: with past-time operators, automata-based operators, propositional quantification, etc. At the end, we provide a complete axiomatic system for LTL and illustrate it with some derivations.

Chapter 7, 'Branching-Time Temporal Logics', introduces and studies a variety of the most important and expressive branching-time temporal logics extending BML with global temporal operators capturing different reachability properties and with quantifiers over paths/computations. These include, inter alia, the temporal logic of reachability TLR, the more expressive computation tree logic CTL, and the full computation tree logic CTL* combining CTL with LTL, plus several fragments, variations and extensions of it. We establish the tree-model property for CTL*, extend the bisimulation invariance result to it and discuss the expressiveness of the family of branching-time logics. In particular, we show that formulae of TLR suffice to characterise every finite interpreted transition system up to bisimulation equivalence. We then study the logical decision problems for CTL and CTL*. We present the linear-time labelling algorithm for model checking CTL formulae and show how model checking in CTL* can be reduced to repeated model checking of LTL formulae. Lastly, we briefly present complete axiomatic systems for TLR and CTL and illustrate them with some derivations.

Chapter 8, 'The Modal Mu-Calculus', presents the most expressive of all temporal logics studied in this book. The modal μ-calculus $\mathcal{L}_\mu$ extends BML with the fundamental syntactic construct of a least fixpoint operator and its dual, the greatest fixpoint operator. Using these, all previously studied temporal operators in the linear and branching-time logics can be defined simply and elegantly. The chapter provides a detailed and technically involved

exposition of the key syntactic and semantic concepts of the modal μ-calculus, including the technical machinery needed to understand and evaluate its formulae, viz. approximants, signatures and games. We present the embedding of all previously studied (single-action) temporal logics, including CTL*, into the μ-calculus, as well as related topics such as modal equation systems, model-checking games and results about structural complexity of formulae. The chapter also offers a comparison with other logical formalisms, such as monadic second-order logic and linear-time μ-calculus.

Chapter 9, 'Alternating-Time Temporal Logics', introduces concurrent game structures as multiagent generalisation of the transition systems considered so far and the respective multiagent extensions of the branching-time logics, known as alternating-time temporal logics, which enable strategic reasoning in open and multiagent transition systems and have recently been gaining increasing popularity. Formulae in the alternating-time temporal logics can quantify over strategies of agents and over all computations consistent with a given collective strategy of a coalition of agents, which is a refinement of the path quantification in CTL*, thus enabling the expression of properties of the type '*the group of agents C has a collective strategy to achieve a given (LTL-definable) temporal objective on all computations consistent with that strategy*'. Following the general structure of the previous chapters, we present the syntax and semantics of the multiagent analogues ATL and ATL* of the branching-time logics CTL and CTL*, discuss their expressiveness and study their logical decision problems. In particular, we show that the linear-time labelling algorithm for model checking CTL extends smoothly to ATL. We also generalise the notion of bisimulation to alternating bisimulation and extend the bisimulation invariance property to ATL. This chapter is relatively independent from the rest of the book, as we do not study alternating-time temporal logics further, but it provides sufficient material for their further study.

Part III, 'Properties', is dedicated to two fundamental generic questions about temporal logics: their expressiveness and the computational complexity of their main logical decision problems. We provide a uniform treatment by proposing a synthetic and unified approach to both questions. These topics are also discussed in the rest of the book, but in a less systematic way. A summary of Part III follows.

Chapter 10, 'Expressiveness', provides an in-depth study of the relationships and comparisons between the different temporal logics introduced in Part II with respect to the temporal properties of interpreted transition systems that can, or cannot, be expressed by formulae in a given particular logic. A way to address such questions is to look at the entire spectrum of all temporal logics. We can naturally compare the expressiveness of two logics by asking whether every property that is formalisable in one of them can also be formalised in the other. This yields a preorder on all temporal logics of the same kind, i.e. with semantics based on the same class of models. A temporal formula can be identified with the class of (rooted) interpreted transition systems in which that formula is valid. Thus, a formula of

one logic being expressible by a formula in another logic simply amounts to the two formulae being equivalent in the usual logical sense, i.e. being valid in the same class of models.

The structure of this chapter reflects the two main kinds of results that are presented there. The first two sections contain positive results of expressive inclusions, by means of translations from one logic to another, where the two different families of logics – for linear time and for branching time – are treated in separate sections. For instance, we show that all the different major extensions of LTL introduced earlier – with automata-based operators (ETL), with propositional quantification (QLTL), the linear μ-calculus LT_μ and the industrial standard Property Specification Language (PSL) – have the same expressiveness. With regards to branching-time logics, we present, for instance, the embedding of CTL* into $\mathcal{L}_\mu$. The last part of Chapter 10 presents negative expressiveness results of the kind that some properties in one logic are not expressible in another. For instance, we show that LTL is expressively weaker than its extensions mentioned earlier, pinpoint the expressive power of CTL by separating it from the other branching-time logics and prove that higher degree of fixpoint alternation in $\mathcal{L}_\mu$ yields greater expressiveness.

Chapter 11, 'Computational Complexity', is devoted to analysing the computational complexity of the main decision problems for temporal logics, which is of great importance for the assessment of their practical suitability as tools for formal verification. The chapter proposes a unifying picture of the computational complexity of most of the temporal logics studied here, by characterising the complexities of their satisfiability and model-checking problems. Complexity upper bounds for these problems are obtained by providing respective decision procedures in the other chapters, by using either concretely described algorithms presented there or by developing general decision methods such as semantic tableaux, automata or games. These upper bounds are revisited in that chapter and matching complexity lower bounds are established there, thus establishing the optimal complexities of the respective decision problems. In particular, the chapter presents a hierarchy class of problems, namely tiling problems and their variants involving games, thus providing a uniform and powerful framework for obtaining lower bound complexity results.

In the last part, **Part IV, 'Methods'**, we present three fundamental methods for solving decision problems for temporal logics, namely: the tableaux-based, the games-based and the automata-based methods, by devoting a chapter to each of these. These technical frameworks are closely related and we discuss briefly their relationships and compare their pros and cons in the brief opening **Chapter 12, 'Frameworks for Decision Procedures'**, provided to help the understanding of the bridges between these methods. Here is a brief summary of the other chapters in this part.

Chapter 13, 'Tableaux-Based Decision Methods', provides a systematic and uniform exposition of (a suitable adaptation) of the method of semantic tableaux, for constructive satisfiability testing and model building for formulae of the logics BML, LTL, TLR and CTL, by organising a systematic search for a satisfying model of the input set of formulae.