

Preface

In classical logic—that is, in propositional logic and in first-order logic—, formulas are either true or false. Modal logic takes a closer look and allows one to distinguish formulas that are true but could be false from formulas that are not only true, but necessarily so. In the same vein, it allows one to distinguish formulas that are false but possibly true from formulas that are not only false, but necessarily so, i.e., impossible.

Aristotle in his *Organon* made exactly the above distinctions, and the concepts of necessity and possibility have interested philosophers since. However, there was no clear and simple semantics before the end of the 1950s. At that time, Saul Kripke popularised the idea that necessity of a formula should be interpreted as *truth of that formula in all possible worlds*; and likewise, possibility of a formula as *truth in some possible world* [Kri63]. Kripke’s idea can be traced back to Leibniz and more recently Carnap. His groundbreaking contribution was to refine that interpretation: he proposed that a formula A has to be evaluated relative to a given possible world w and that A is necessarily true at w if and only if A is true in all possible worlds *that are accessible from w* . So there might be worlds that are possible but inaccessible. While similar proposals were independently made by Marcel Guillaume, Stig Kanger, Jaakko Hintikka, and others, possible worlds semantics became known under the name “Kripke semantics”.

What are possible worlds? For a philosopher this is a subject of debate, with issues such as: do they really exist? what kind of possibility are we talking about? etc. The picture is clear in theoretical computer science: possible worlds are viewed as states in which a computer program might be, and the relation of accessibility is viewed as a *transition relation between states*. Quite differently, in semantic web ontologies possible worlds are objects from some domain, and the accessibility relations are *relations between objects*. Possible worlds are *nodes in a graph*: node w_2 is accessible from node w_1 via the accessibility relation if there is an edge (an arrow) from w_1 to w_2 that is labelled by that relation.

Beyond Kripke’s original formal analysis of the concepts of necessity and possibility, possible worlds models quickly turned out to be a flexible tool to investigate a whole family of other concepts.

- Temporal concepts such as ‘always’ and ‘sometimes,’ ‘henceforth’ and ‘eventually,’ ‘next,’ ‘until’ and ‘since,’ ‘before’ and ‘after’ can be naturally interpreted with accessibility relations. These relations are typically orders, in particular linear orders. This was first proposed by Arthur Prior [Pri57, Pri67].
- What is necessarily or possibly true after the execution of a (possibly nondeterministic) computer program π can be described by associating transition relations between states to π . This was initiated by Vaughn Pratt, after earlier work by Andrzej Salwicki [Pra76].
- Knowledge and belief of an agent I can be interpreted as truth in all worlds that are possible for I . This was first proposed by Jaakko Hintikka [Hin62].
- Deontic concepts such as obligation and permission can be interpreted as truth in all (resp. some) ideal worlds. Building on earlier work by G.H. von Wright [vW51], such an analysis was advocated by Stig Kanger and Jaakko Hintikka.
- Motivational concepts such as goals and intentions of an individual I can be interpreted as truth in all worlds that are preferred by I . This was studied among others by Philip Cohen and Hector Levesque [CL90a, CL90b].
- Conditionals of the form “If A then B ” were interpreted as truth of B in all those worlds where A is true and that are closest to the actual world. This was studied by Robert Stalnaker and David Lewis [Sta68, Lew73].

All these concepts are called *modalities*. That is the wide sense of the term; in the narrow sense, there are only two modalities: necessity and possibility.

When investigating the modal logic of a particular concept such as belief one typically starts by analysing the *logical form* of that concept. For example, the object of an agent’s belief is a formula, and an agent’s belief that a formula is true is itself a formula. One then defines a *logical language* containing one or more modal operators: a single operator of belief if one only considers one agent, or several operators of belief, one per agent. The formulas of that language can then be interpreted in terms of possible worlds and accessibility relations between possible worlds. Each particular concept one wishes to investigate has specific properties. For example, the temporal concept of a formula being eventually true is interpreted by an accessibility relation that is both reflexive and transitive: if A is true now then it can be said that A is eventually true; and if it is eventually true that A is eventually true, then A is eventually true; in contrast, the temporal ‘next’ (‘at the next state’) relation is neither reflexive nor transitive.

To design a logic amounts to identifying the list of constraints that the accessibility relations should satisfy. Formulas of a modal language are therefore interpreted in particular *classes of models*. For example the language with the modal operator ‘eventually’ is interpreted in the class of Kripke models whose accessibility relation for ‘eventually’ is reflexive and transitive.

In all these proposals, possible worlds are viewed as complete descriptions of a state of affairs. In a different tradition, possible worlds are viewed as objects of the world (just as in first-order logic), and the formula A is viewed as a property of the object under concern. Then ‘necessarily A ’ expresses a restricted quantification: all objects that are related to the object currently under concern have property A . This view leads to the definition of knowledge representation languages that generalise

relational databases and semantic networks. Their advantage over first-order logic is that—just as many other modal logics—they have good mathematical properties. They are in particular said to be “Robustly decidable” [Var96]: More precisely, it is not only decidable whether a given formula A is true in a given world of a given Kripke model, but also whether for a given formula A , a given class of Kripke models contains a model such that A is true in some world of that model. The former reasoning problem is called the model checking problem; the latter is called the satisfiability problem.

Why Did We Write This Book?

Why did we choose to write this book? Several textbooks on modal logics already exist, both at the introductory and at the more advanced level.

Almost all introductory texts start with syntax and semantics, and then focus on Hilbert-style axiomatizations. Examples of such books are the classical [HC68, HC84] and [Che80], and the more recent [CP09]. All of these devote relatively little attention to the above-mentioned reasoning tasks of model checking and satisfiability checking, nor are they geared towards automatic procedures. Moreover, the first three of them are almost exclusively about *monomodal* logics: modal logics having only one modal operator of necessity and one modal operator of possibility. They do not account for logics that are about more than one concept, such as logics of knowledge and action, logics of knowledge and obligation, or logics of belief and time.

There exist more advanced textbooks that cover not only semantics and Hilbert axiomatizations, but also currently known decidability and complexity results. Among these books there are [CZ97, BdRV01, GKWZ03, BBW06, GSS03]. However, all these books contain quite demanding material going far beyond introductory textbooks. The same can be said for textbooks about particular modal logics such as the logic of programs [HKT00], the logics of time [FGV05], or the logics of ontologies [BCM+03].

The present book aims at filling this gap: our aim is to introduce the reader to the most important modal logics with multiple modalities, and we would like to do so from the perspective of the associated automated reasoning tasks. To that end we concentrate on the most general and powerful reasoning method for modal logics: tableaux systems.

The central idea of the tableaux method is: try to build a Kripke model for a given formula by applying the truth conditions. More than 40 years ago, Melvin Fitting showed how to extend tableaux systems from classical logic to modal logics. He used these systems in his textbook order to introduce the basic modal logics in a systematic way [Fit83]. It has to be noted that Fitting exclusively treated monomodal logics; moreover, the tableaux systems were presented in the way they were supposed to be used at that time, namely in order to write down proofs with paper and pencil.

Since the 1990s there has been a trend to design and implement tableaux systems on computers. Prominent examples are the Logic Workbench LWB [HSZ96] and several tableaux provers for description logics [TH06]. For an up-to-date overview we refer the reader to a webpage that is maintained by Renate Schmidt at the University of Manchester.¹ Our motivation to write this book was to use implemented tableaux systems in order to provide a way to gently introduce the reader to the wealth of currently existing modal logics. However, the above-mentioned implementations are all dedicated to either just one modal logic, or to a small family thereof. In contrast, a general introduction to modal logics requires a general and versatile piece of software that is able to account for the numerous existing modal logics. That set being infinite, what is needed is a generic tool that can be easily instantiated in order to implement a given modal logic—possibly a new logic that had never been considered before. There exist only three generic tableaux provers: the Tableaux Workbench TWB² [AG09], the Tableau Prover Generator MetTel2³ [TSK12] and the vehicle of our book: LoTREC.

The tableaux method is usually presented in a way that comes very close to that of Gentzen sequent systems. The latter can be used to build proofs that take the form of trees, breaking down a given input formula connective by connective. Such sequent systems can be defined quite straightforwardly for many basic modal logics; by and large, those whose models can be identified with trees. However, Kripke models are not limited to trees, and both sequent and tableaux systems for logics with e.g. symmetric accessibility relations are more difficult to design: somewhat in opposition with Gentzen’s spirit they typically require side conditions, leading to a cumbersome meta-linguistic machinery. Contrasting with almost all existing tree-based tableaux systems, we are going to work on graphs in order to get around that difficulty. Our presentation of tableaux is therefore much closer to Kripke models.

Staying close to Kripke models has a second advantage: it allows one to introduce at the same time the semantics of modal logics on the one hand, and tableaux proof systems on the other.

There is another difference with traditional presentations of tableaux systems: just as in the case of sequent systems, most of the explanations are in terms of the search for a proof of validity of a formula that is done by refuting its negation. In contrast, we focus on the construction of a model for the input formula: for us, a tableaux system is a *model construction* procedure.

The price to pay for generality and simplicity of use is lack of efficiency: while LoTREC performs reasonably well on the basic modal logics, it is outperformed by more specialized tableaux implementations... when run on the logics the latter are designed for. In contrast, when one wants to implement a new tableaux system then LoTREC provides a simple and generic language to do so, while existing implementations are difficult to adapt and require diving into the programming language code in which the tableau prover is implemented, such as C++ in the case of the LWB.

¹<http://www.cs.man.ac.uk/~schmidt/tools>.

²<http://twb.rsise.anu.edu.au>.

³<http://www.mettel-prover.org>.

To sum it up, the aim of this book is to provide a gentle introduction to multi-modal logics via tableaux systems. It should enable readers that do not have any knowledge beyond propositional logic to learn what modal logics are about, which properties they have and how they can be put to work; in particular, we hope that it will allow them to implement and play with existing or new modal logics. Most importantly, they should be able to do so *without any knowledge or skills in programming*. Our tool in this enterprise is LoTREC. In each of the chapters in the book the reader may turn into a LoTREC user in order to actively explore the functioning and the programming of tableaux systems.

LoTREC: An Ubiquitous Tool in This Book

LoTREC is a free software that is distributed by the *Institut de recherche en informatique de Toulouse* (IRIT) and that is accessible at the following webpage:

<http://www.irit.fr/Lotrec>

While the name LoTREC refers to the French painter Henri de Toulouse Lautrec, it can also be understood as the acronym of the—admittedly clumsy—“**Logical Tableaux Research Engine Companion**.”

The language of LoTREC has three basic concepts: logical connectives, tableau rules and strategies.

- The user of LoTREC can define his own logical language by defining logical connectives of any arity: one, two, or more.
- A *tableau rule* is of the form “If *condition* then *action*” and enables construction of Kripke models.
- A *strategy* describes in which order the rules are applied.

LoTREC is *generic*: by defining unique connectives, rules and strategies the LoTREC user can implement a tableau proof procedure for the logic of interest. This is supported by a user-friendly graphical interface. LoTREC should therefore ease both the task of students in learning and the task of researchers in debugging and prototyping.

LoTREC can either be run on-the-fly via the web, or be installed on the user’s computer. It is implemented in Java and is therefore cross-platform: it can be run under operating systems such as Windows, Unix, Macintosh, and Solaris.

The implementation of LoTREC was done in the framework of several Master and PhD theses. It started with David Fauthoux’s 2000 MSc thesis, that was based on ideas from a 1997 Fundamenta Informaticae paper by Marcos Castilho, Luis Fariñas del Cerro, Olivier Gasquet and Andreas Herzig [CFdCGH97]. Mohamad Sahade’s 2006 and Bilal Said’s 2009 PhD theses worked things out in detail, both in theory and implementation.

Overview of the Chapters

The first two chapters introduce Kripke models and the language of modal logic. The rest of the chapters present the tableaux method for various families of modal logics.

Chapter 1. Modelling with Graphs In the first chapter we introduce Kripke models. With the help of a series of examples we present the main modal concepts that can be interpreted by means of possible worlds semantics: the concepts of event, action, program, time, belief, knowledge, and obligation. In the end of the chapter we show how Kripke models can be built by means of LoTREC and give a formal definition of Kripke models.

Chapter 2. Talking About Graphs In the second chapter we introduce a general formal language in order to talk about properties of Kripke models. We then show how such languages can be defined in LoTREC and how to check that a formula is true in a given world of a given Kripke model. The latter is called model checking. Beyond model checking we also introduce the reasoning tasks of satisfiability checking, validity checking, and model building. We show that all other tasks can be reduced to the latter, which we focus on in the rest of the book: we show how to build models for a series of logics. These logics are grouped into families, according to the techniques the tableaux implementation in LoTREC requires.

Chapter 3. The Basics of the Model Construction Method This chapter is about the basic modal logic $\mathbf{K}$ and its multimodal version $\mathbf{K}_n$. We also present the basic description logic $\mathbf{ALC}$, which can be viewed as a notational variant of $\mathbf{K}_n$. The implementation of reasoning methods for all these logics can be done by means of the most basic rules, that are combined by the most basic strategies: fair strategies.

Chapter 4. Logics with Simple Constraints on Models This chapter is about the model construction problem in some classes of models: models satisfying the conditions of reflexivity, seriality, symmetry, and combinations thereof. The corresponding logics are $\mathbf{KT}$, $\mathbf{KD}$, $\mathbf{KB}$, $\mathbf{KTB}$, and $\mathbf{KDB}$. We also consider models whose accessibility relation is confluent (logic $\mathbf{K.2}$) or is an equivalence relation (logic $\mathbf{KT45}$, alias $\mathbf{S5}$). While these conditions are only about a single relation, we also study properties involving two accessibility relations: inclusion and permutation. The reason for grouping these classes of models in a chapter is that model construction can be implemented by means of the most basic rules. We therefore call all these constraints *simple*. In the end of the chapter we present a general termination theorem that holds for almost all of these logics and which guarantees that the tableau construction does not loop.

Chapter 5. Logics with Transitive Accessibility Relations This chapter is about the model construction problem in classes of models satisfying the constraint of transitivity. We present the modal logics of the class of models where the accessibility relation is transitive ($\mathbf{K4}$), transitive and serial ($\mathbf{KD4}$), and transitive and

reflexive (**KT4**, alias **S4**). For these logics, the model construction procedure may loop, which contrasts with the simple logics of Chap. 4. Termination can be ensured by means of blocking techniques: basically, the construction is stopped when the labels of a node are identical to those of some ancestor node. In the end of the chapter we present another general termination theorem guaranteeing that the tableau construction does not loop and which applies to all these logics.

Chapter 6. Model Checking This chapter shows how to implement model checking in **LoTREC**. There are two reasons why this topic is placed here: first, the implementation of model checking in **LoTREC** requires us to extend the tagging primitives of Chap. 5; second, model checking is going to be used in the next chapter.

Chapter 7. Modal Logics with Transitive Closure This chapter is about the model construction problem in classes of models having relations that are transitive closures of other relations. The main such logics are linear-time temporal logic **LTL** and propositional dynamic logic **PDL**. These logics require both blocking and model checking.

As we have already pointed out, this book not only presents the most important modal logics, but also tableaux proof procedures for them. We start by giving the **LoTREC** primitives for building models (Chap. 1) and for defining the language (Chap. 2). In an incremental way, each of the following chapters introduces the **LoTREC** primitives that are needed in order to automatically build models for the logics it is about.

Audience and Prerequisites

Readers of this book need only little background in mathematics and logic: some knowledge of the bases of propositional logic is sufficient. It is intended for undergraduate students ranging from the humanities and social sciences to computer science and mathematics.

Acknowledgments

Thanks are due to several persons who helped us to launch the idea and to improve this book. This is the place to warmly thank all of them.

The members of the Logic, Interaction, Language and Computation group (LILaC) at the Institut de Recherche en Informatique de Toulouse (IRIT) provided support and feedback throughout the years. David Fauthoux and Mohamad Sahade respectively wrote their Master and PhD thesis on **LoTREC**. Philippe Balbiani, Luis Fariñas del Cerro, Dominique Longin, and Emiliano Lorini gave us precious advice. Philippe Balbiani, Teddy Bouziat, Emiliano Lorini, and Faustine Maffre tested

LoTREC and read parts of the manuscript, allowing us to correct several mistakes. LILaC, IRIT and Toulouse are wonderful places to be and to work!

The material in the present book was presented partly or entirely in several workshops, conferences, and tutorials at spring and summer schools, in particular at the 2009 Universal Logic (UNILog) spring school and at the 2009 European Summer School in Logic, Language and Information (ESSLI 2009). The questions, remarks and comments of the attendees allowed us to improve the presentation.

Finally, many users of LoTREC—several of which used LoTREC when teaching introduction to modal logic classes—sent us encouraging feedback and suggested some improvements. To get an email saying that LoTREC was simple and easy to use was always a pleasure for us!

Toulouse, France
July 2013

Olivier Gasquet
Andreas Herzig
Bilal Said
François Schwarzentruher

Kripke's Worlds

An Introduction to Modal Logics via Tableaux

Gasquet, O.; Herzig, A.; Said, B.; Schwarzentruher, F.

2014, XV, 198 p. 73 illus., Softcover

ISBN: 978-3-7643-8503-3

A product of Birkhäuser Basel